US009542334B2

(12) **United States Patent**
Ramaraju

(10) **Patent No.: US 9,542,334 B2**
(45) **Date of Patent: Jan. 10, 2017**

(54) **MEMORY MANAGEMENT UNIT TAG MEMORY WITH CAM EVALUATE SIGNAL**

(75) Inventor: **Ravindraraj Ramaraju**, Round Rock, TX (US)

(73) Assignee: **NXP USA, INC.**, Austin, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1208 days.

(21) Appl. No.: **13/213,831**

(22) Filed: **Aug. 19, 2011**

(65) **Prior Publication Data**

US 2013/0046927 A1 Feb. 21, 2013

(51) **Int. Cl.**
G06F 12/10 (2016.01)
G06F 9/38 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 12/1027*** (2013.01); ***G06F 9/3824*** (2013.01)

(58) **Field of Classification Search**
None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,754,819	A	5/1998	Lynch et al.	
6,088,763	A	7/2000	Silberman et al.	
6,292,818	B1	9/2001	Winters	
6,466,960	B1	10/2002	Winters	
6,813,628	B2	11/2004	Bhushan et al.	
6,865,590	B2	3/2005	Park	
6,957,315	B2	10/2005	Chauvel	
7,035,127	B1	4/2006	Adams et al.	
7,669,034	B2	2/2010	Bearden et al.	
7,984,229	B2	7/2011	Ramaraju et al.	
8,099,580	B2	1/2012	Ramaraju et al.	
8,380,779	B2	2/2013	Bruce et al.	
2005/0050278	A1	3/2005	Meier et al.	
2007/0028051	A1	2/2007	Williamson et al.	
2007/0033318	A1	2/2007	Gilday et al.	
2007/0094480	A1 *	4/2007	Ramaraju et al.	711/220
2010/0306302	A1 *	12/2010	Ramaraju et al.	708/670

OTHER PUBLICATIONS

T. Juan et al., Reducing TLB Power Requirements, International Symposium on Low Power Electronics and Design, Aug. 18-20, 1997, pp. 196-201.

M.T. Vaden et al., Design Considerations for the PowerPC 601 Microprocessor, IBM Journal of Research Development, vol. 38, No. 5, Sep. 1994.

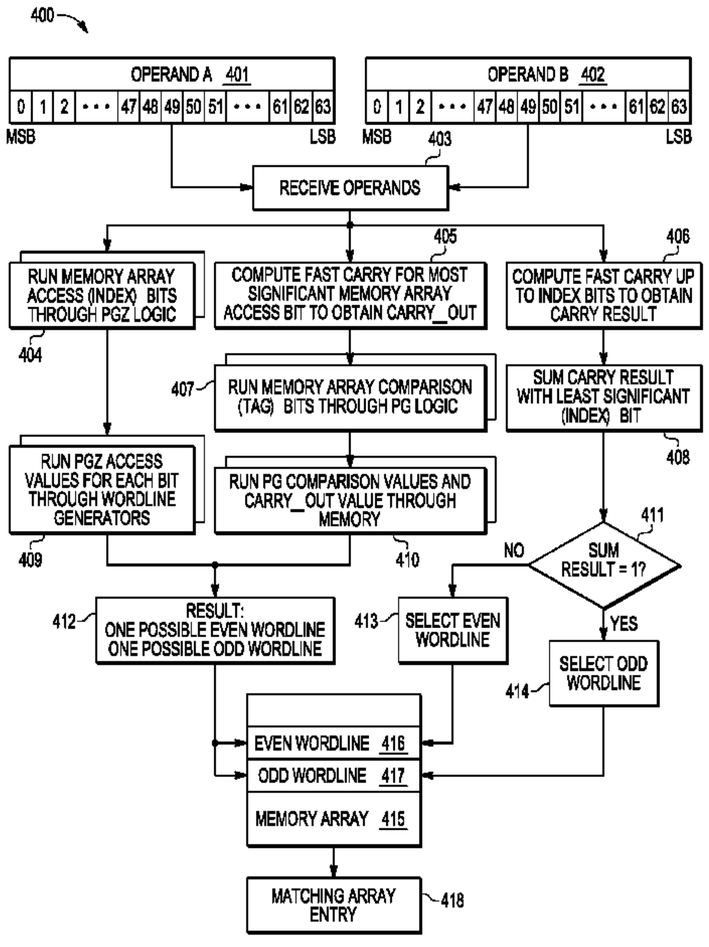
(Continued)

Primary Examiner — Gary W Cygiel

(57) **ABSTRACT**

A method and data processing system for accessing an entry in a memory array by placing a tag memory unit (**114**) in parallel with an operand adder circuit (**112**) to enable tag lookup and generation of speculative way hit/miss information (**126**) directly from the operands (**111**, **113**) without using the output sum of the operand adder. PGZ-encoded address bits (**0**:**51**) from the operands (**111**, **113**) are applied with a carry-out value ($Cout_{48}$) to a content-addressable memory array (**114**) having compact bitcells with embedded partial A+B=K logic to generate two speculative hit/miss signals under control of a delayed evaluate signal. A sum value (EA_{51}) computed from the least significant base and offset address bits determines which of the speculative hit/miss signals is selected for output (**126**).

20 Claims, 9 Drawing Sheets



(56) **References Cited**

OTHER PUBLICATIONS

Cortadella et al., "Evaluation of A+B=K Conditions w/o Carry Propagation", IEEE Transactions on Computers, Nov. 1992.
Heald, R. et al., "64 kB sum-addressed-memory cache with 1.6 ns cycle and 2.6 ns latency," IEEE JSSC, vol. 33, Issue: 11, 1998, pp. 1682-1689.
Austin et al., "Streamlining Data Cache Access w/ Fast Address Calculation", 22nd Intl Symp on Computer Architecture, 1995.
U.S. Appl. No. 13/213,900 entitled "Memory Management Unit Tag Memory", filed on Aug. 19, 2011.
Non-final office action dated Nov. 8, 2013 in U.S. Appl. No. 13/213,900.
Final office action dated Feb. 21, 2014 in U.S. Appl. No. 13/213,900.
Non-final office action dated Jun. 2, 2014 in U.S. Appl. No. 13/213,900.
Final office action dated Nov. 20, 2014 in U.S. Appl. No. 13/213,900.
Notice of Allowance dated Mar. 18, 2015 in U.S. Appl. No. 13/213,900.

* cited by examiner

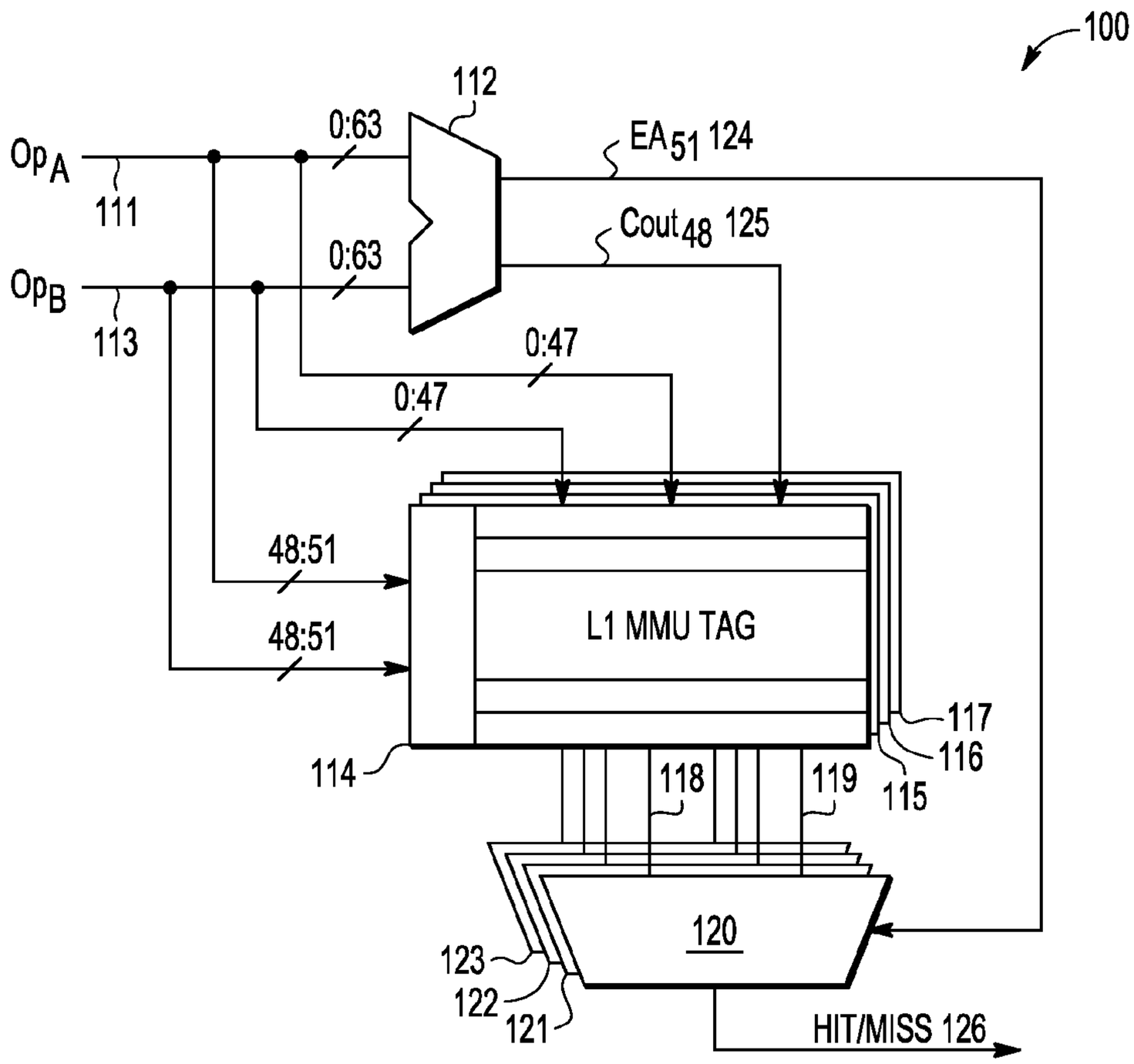

100
112
0:63
Op_A
111
0:63
Op_B
113
EA_{51} 124
$Cout_{48}$ 125
0:47
0:47
48:51
48:51
L1 MMU TAG
114
117
116
115
118
119
120
123
122
121
HIT/MISS 126

FIG. 1

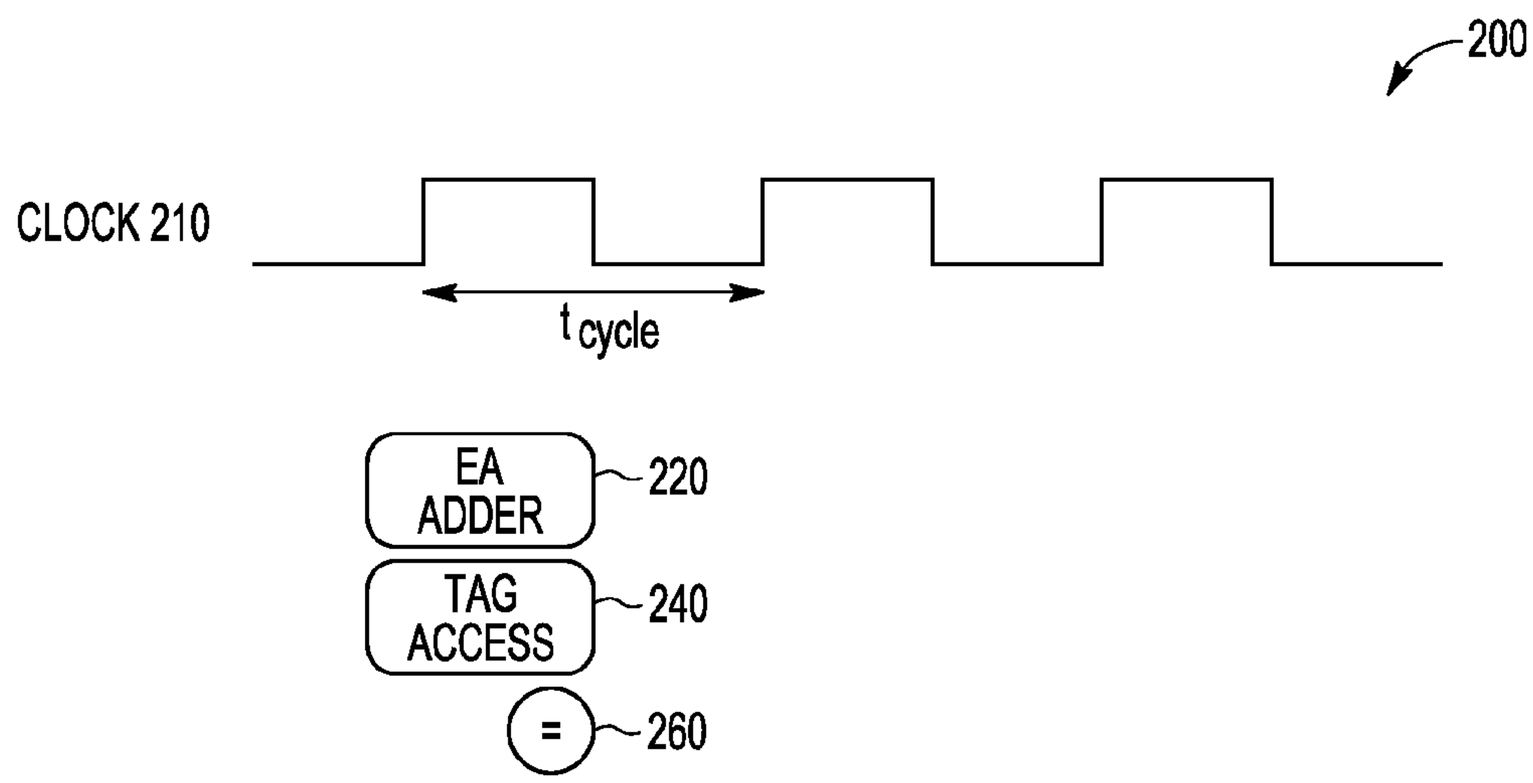

200
CLOCK 210
t_{cycle}
EA ADDER
220
TAG ACCESS
240
=
260

FIG. 2

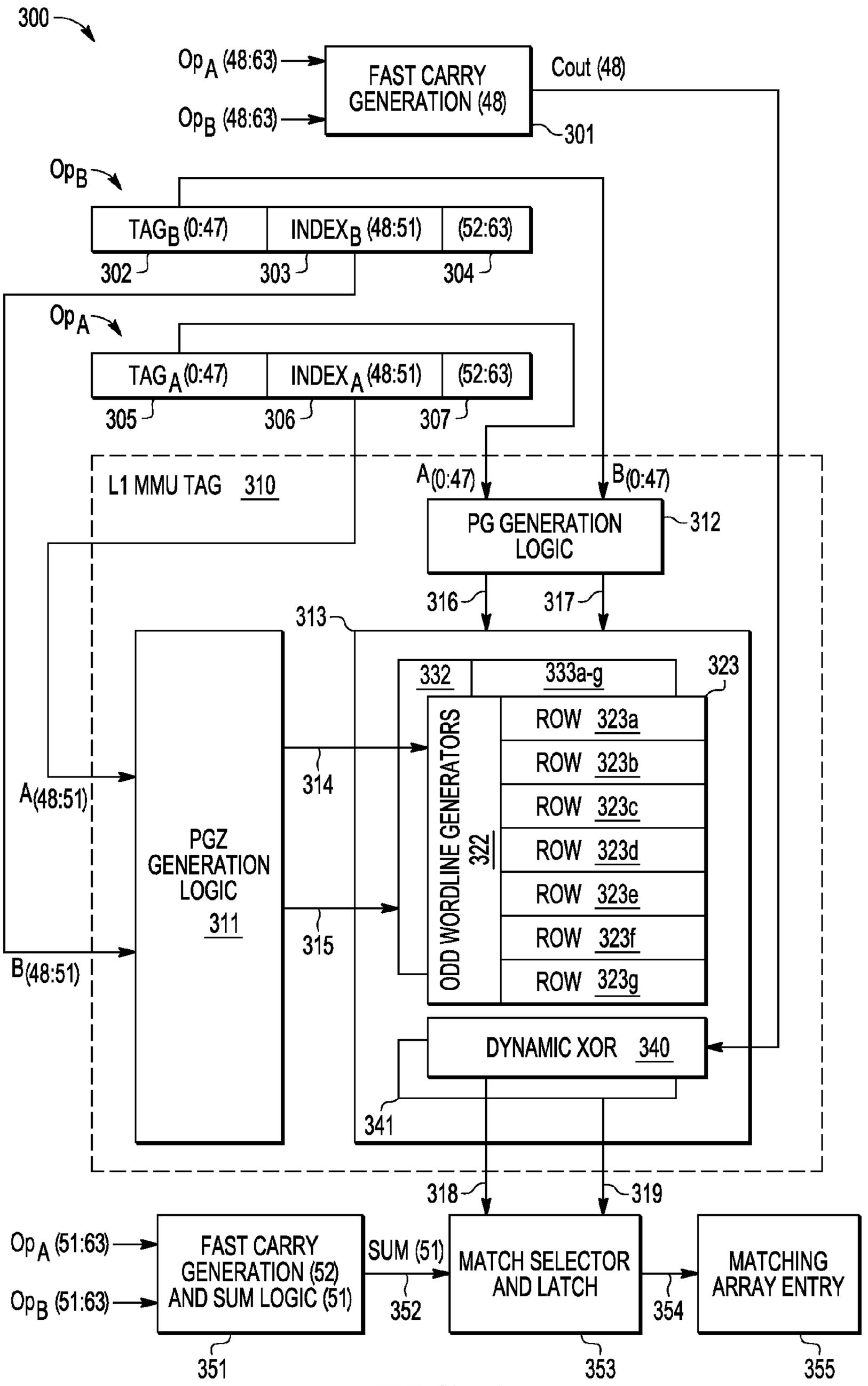

300
Op A (48:63)
Op B (48:63)
FAST CARRY GENERATION (48)
Cout (48)
301
Op B
TAG B (0:47)
INDEX B (48:51)
(52:63)
302
303
304
Op A
TAG A (0:47)
INDEX A (48:51)
(52:63)
305
306
307
L1 MMU TAG 310
A(0:47)
B(0:47)
PG GENERATION LOGIC
312
316
317
313
332
333a-g
323
ODD WORDLINE GENERATORS
322
ROW 323a
ROW 323b
ROW 323c
ROW 323d
ROW 323e
ROW 323f
ROW 323g
314
315
A(48:51)
B(48:51)
PGZ GENERATION LOGIC
311
DYNAMIC XOR 340
341
318
319
Op A (51:63)
Op B (51:63)
FAST CARRY GENERATION (52) AND SUM LOGIC (51)
SUM (51)
352
MATCH SELECTOR AND LATCH
354
MATCHING ARRAY ENTRY
351
353
355

FIG. 3

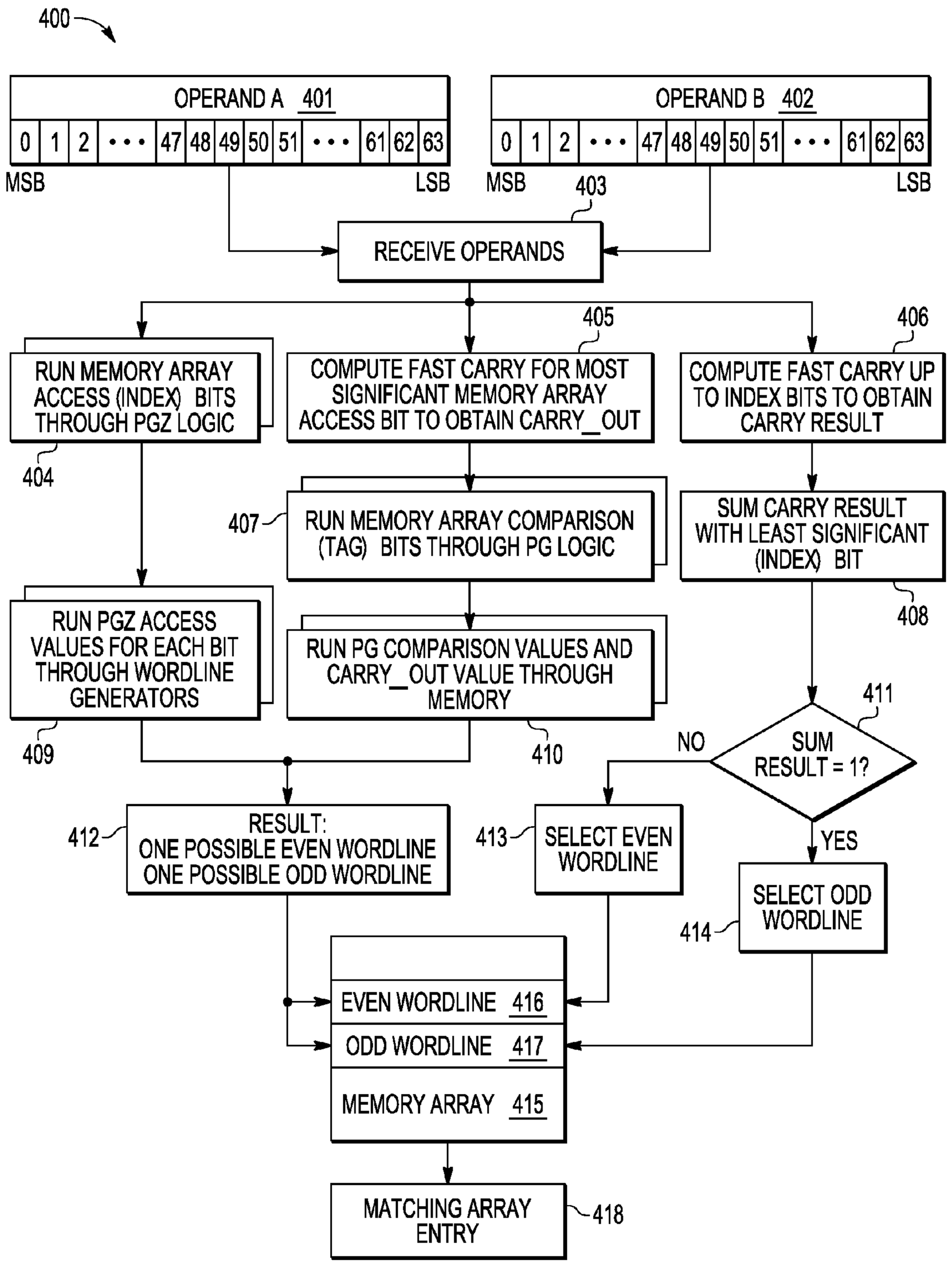

400
OPERAND A 401
0 1 2 • • • 47 48 49 50 51 • • • 61 62 63
MSB
LSB
OPERAND B 402
0 1 2 • • • 47 48 49 50 51 • • • 61 62 63
MSB
LSB
403
RECEIVE OPERANDS
RUN MEMORY ARRAY ACCESS (INDEX) BITS THROUGH PGZ LOGIC
404
COMPUTE FAST CARRY FOR MOST SIGNIFICANT MEMORY ARRAY ACCESS BIT TO OBTAIN CARRY_OUT
405
COMPUTE FAST CARRY UP TO INDEX BITS TO OBTAIN CARRY RESULT
406
RUN MEMORY ARRAY COMPARISON (TAG) BITS THROUGH PG LOGIC
407
SUM CARRY RESULT WITH LEAST SIGNIFICANT (INDEX) BIT
408
RUN PGZ ACCESS VALUES FOR EACH BIT THROUGH WORDLINE GENERATORS
409
RUN PG COMPARISON VALUES AND CARRY_OUT VALUE THROUGH MEMORY
410
411
SUM RESULT = 1?
NO
YES
RESULT: ONE POSSIBLE EVEN WORDLINE ONE POSSIBLE ODD WORDLINE
412
SELECT EVEN WORDLINE
413
SELECT ODD WORDLINE
414
EVEN WORDLINE 416
ODD WORDLINE 417
MEMORY ARRAY 415
MATCHING ARRAY ENTRY
418

FIG. 4

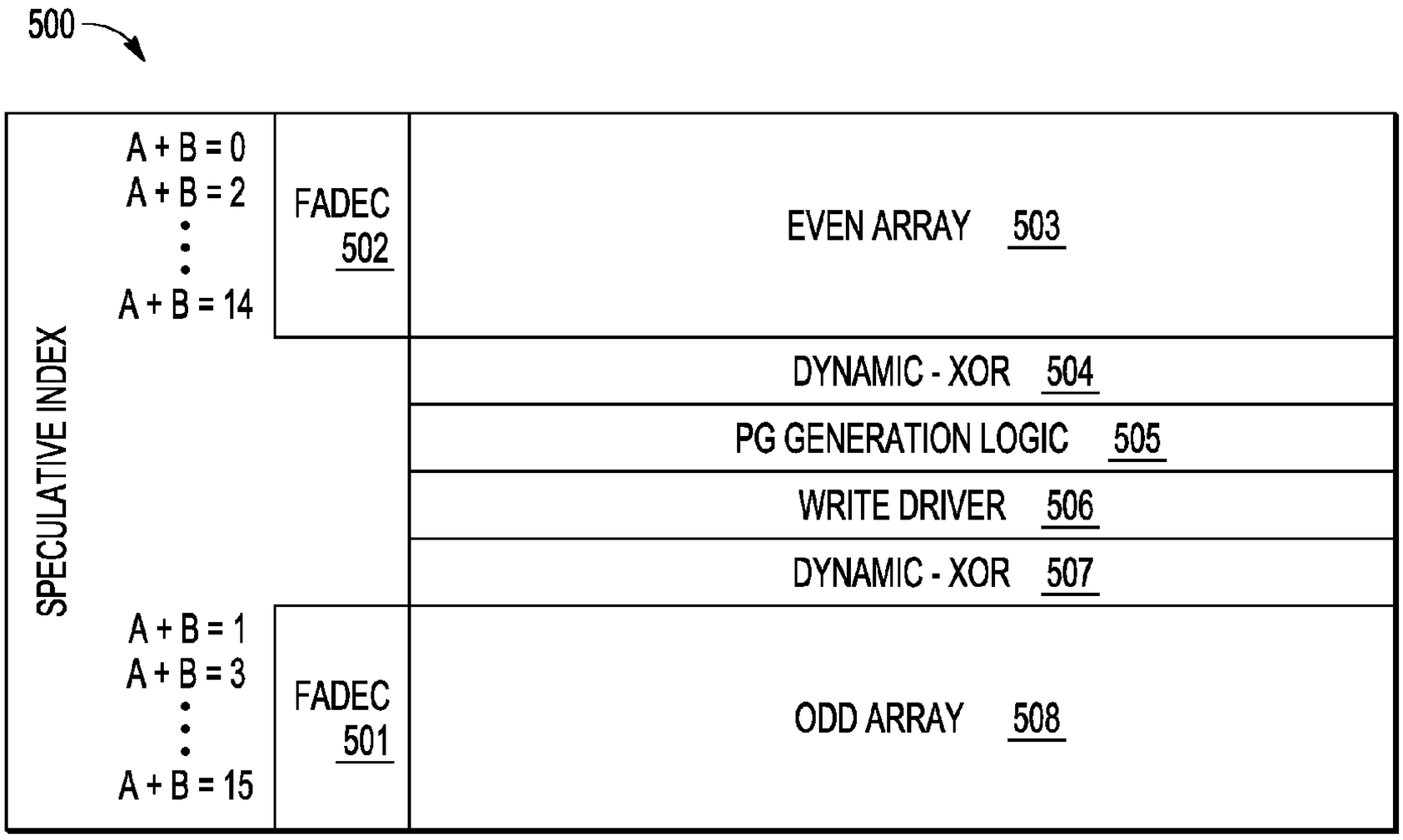

500
SPECULATIVE INDEX
A + B = 0
A + B = 2
A + B = 14
FADEC 502
EVEN ARRAY 503
DYNAMIC - XOR 504
PG GENERATION LOGIC 505
WRITE DRIVER 506
DYNAMIC - XOR 507
A + B = 1
A + B = 3
A + B = 15
FADEC 501
ODD ARRAY 508

FIG. 5

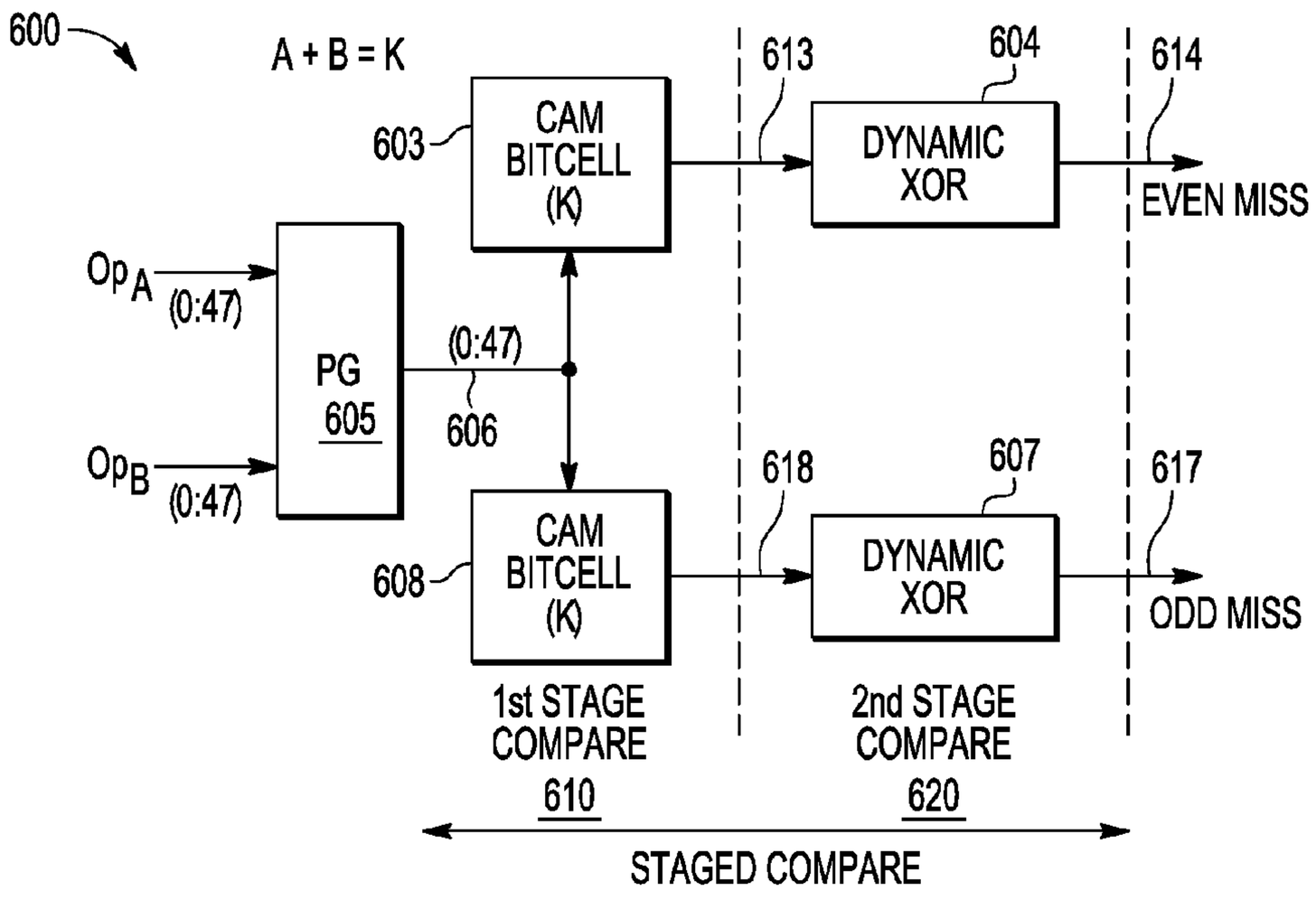

600
A + B = K
603
CAM BITCELL (K)
613
604
DYNAMIC XOR
614
EVEN MISS
Op A (0:47)
Op B (0:47)
PG 605
(0:47)
606
608
CAM BITCELL (K)
618
607
DYNAMIC XOR
617
ODD MISS
1st STAGE COMPARE 610
2nd STAGE COMPARE 620
STAGED COMPARE

FIG. 6

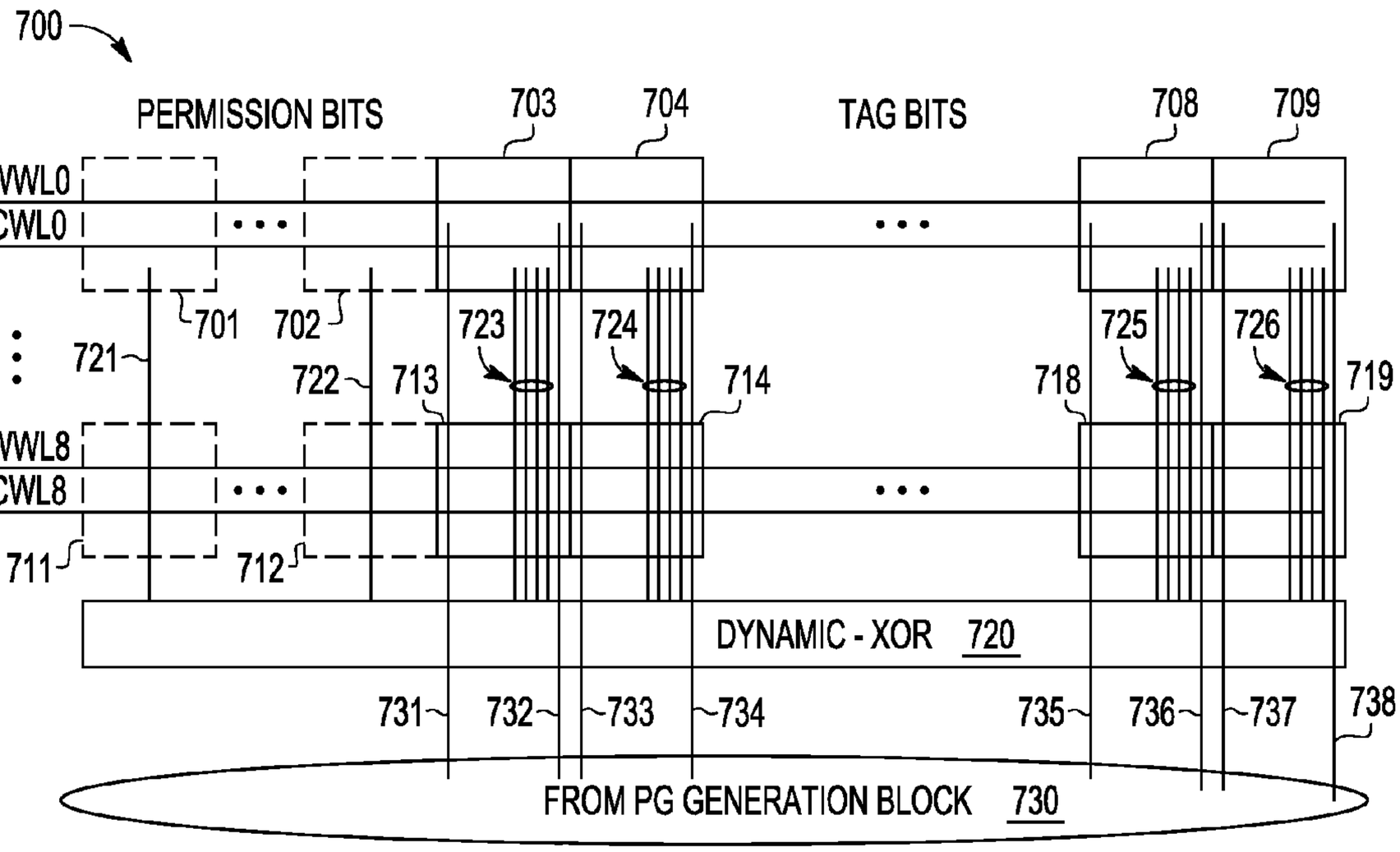

700
PERMISSION BITS
703
704
TAG BITS
708
709
WWL0
CWL0
701
702
723
724
725
726
721
722
713
714
718
719
WWL8
CWL8
711
712
DYNAMIC - XOR 720
731
732
733
734
735
736
737
738
FROM PG GENERATION BLOCK 730

FIG. 7

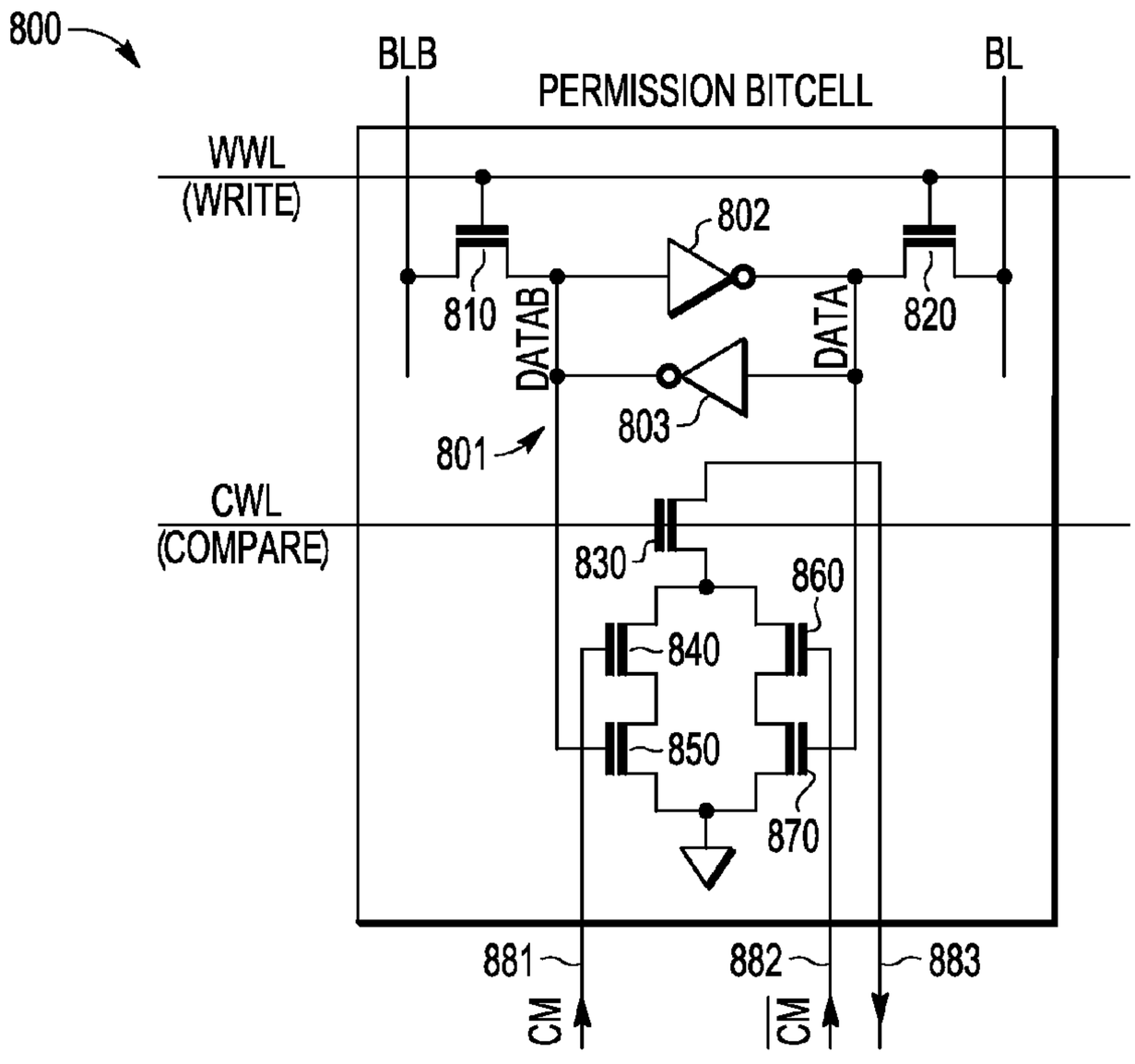

800
BLB
BL
PERMISSION BITCELL
WWL
(WRITE)
802
810
DATAB
DATA
820
803
801
CWL
(COMPARE)
830
860
840
850
870
881
882
883
CM
CM

FIG. 8

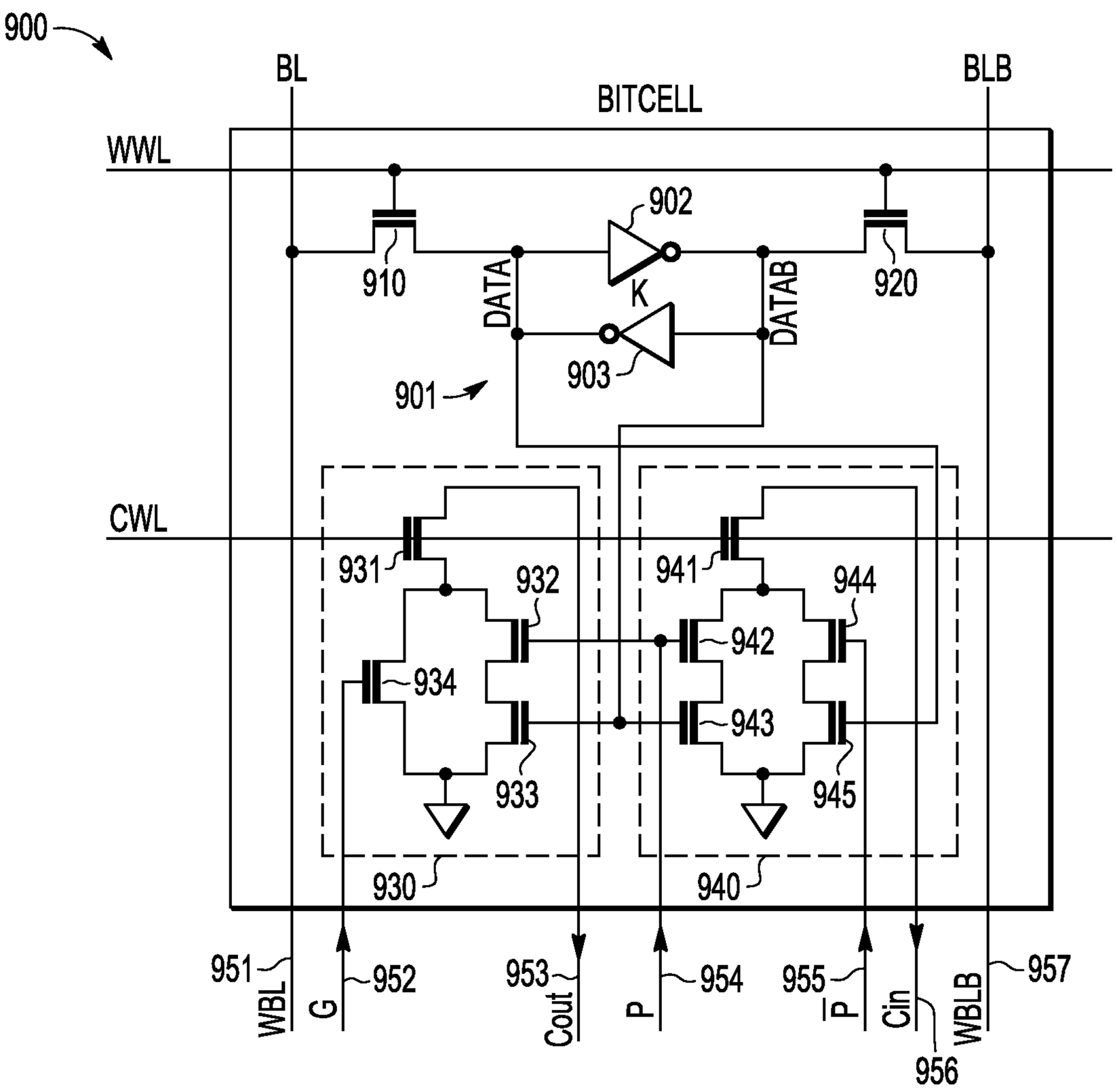
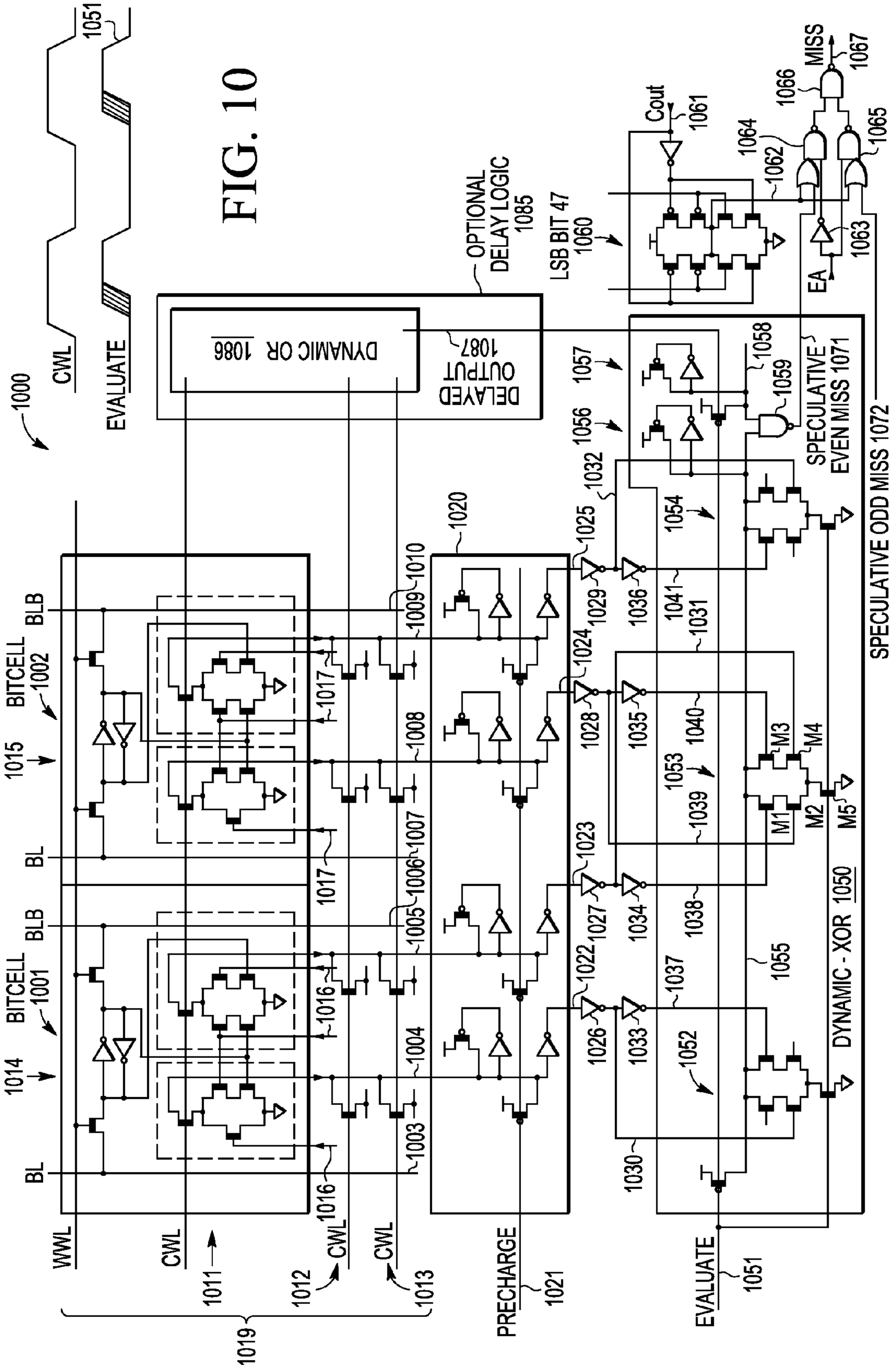

900
BL
BLB
BITCELL
WWL
902
910
DATA
K
DATAB
920
903
901
CWL
931
932
941
944
942
934
943
933
945
930
940
951
WBL
G
952
953
Cout
P
954
955
P
Cin
956
WBLB
957

FIG. 9

FIG. 10


1000
CWL
EVALUATE
1051
DYNAMIC OR 1086
DELAYED OUTPUT 1087
OPTIONAL DELAY LOGIC 1085
LSB BIT 47 1060
Cout 1061
1064
1062
1066
MISS 1067
1065
EA
1063
1057
1056
1058
1059
SPECULATIVE EVEN MISS 1071
SPECULATIVE ODD MISS 1072
1032
1025
1054
1020
1010
1009
1029
1036
1041
1031
1024
1028
1035
1040
1053
1039
M1
M2
M3
M4
M5
1023
1027
1034
1038
1037
1022
1026
1033
1052
1055
1030
1050
DYNAMIC - XOR 1054
BITCELL 1002
1015
BLB
BL
1017
1008
1007
1006
1005
BITCELL 1001
1014
1016
1004
1003
WWL
CWL
1011
1012
1013
1019
PRECHARGE
1021
EVALUATE
1051

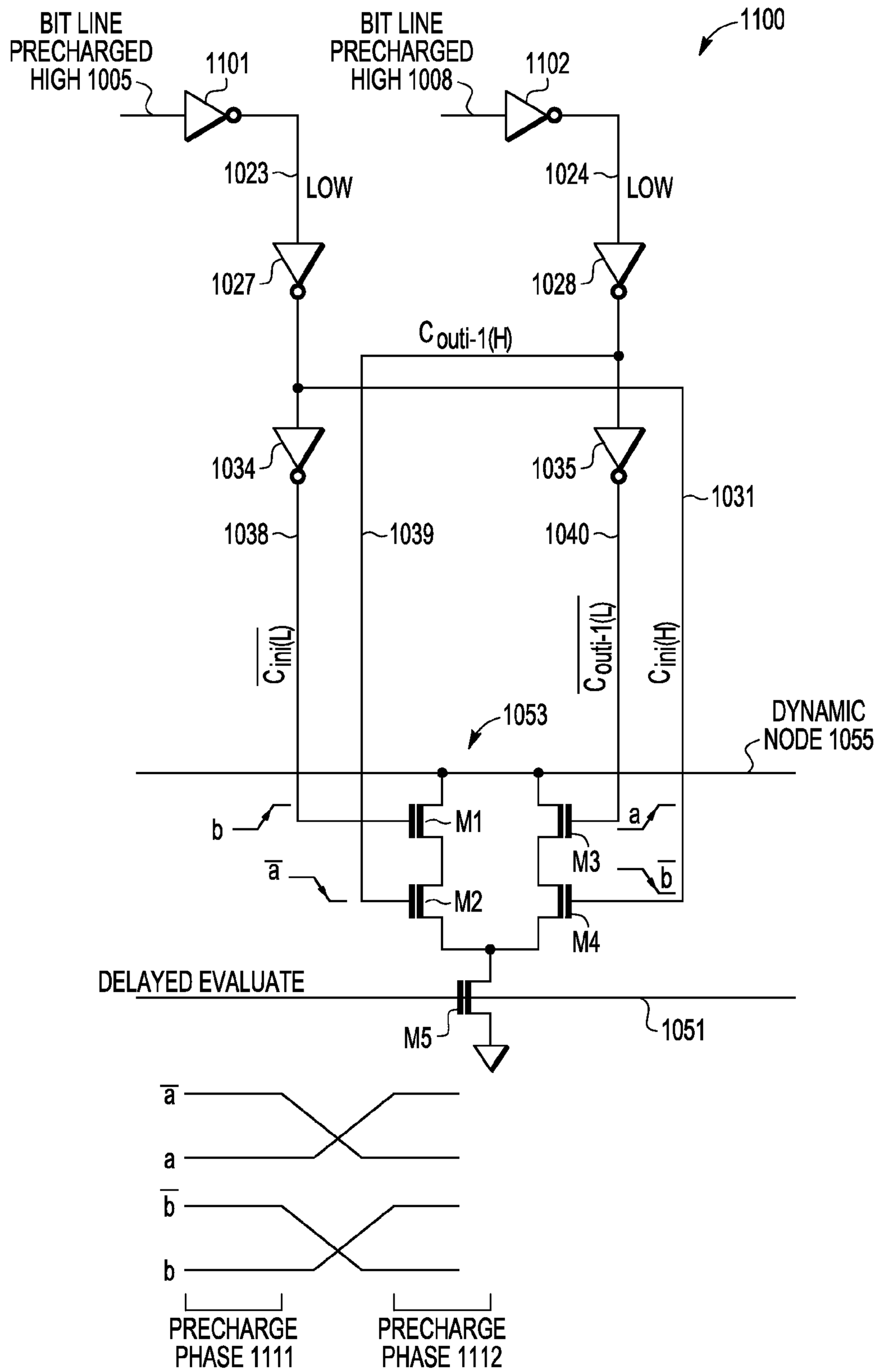

1100
BIT LINE PRECHARGED HIGH 1005
1101
BIT LINE PRECHARGED HIGH 1008
1102
1023
LOW
1024
LOW
1027
1028
C outi-1(H)
1034
1035
1031
1038
1039
1040
C ini(L)
C outi-1(L)
C ini(H)
1053
DYNAMIC NODE 1055
b
M1
a
M3
a
b
M2
M4
DELAYED EVALUATE
M5
1051
a
a
b
b
PRECHARGE PHASE 1111
PRECHARGE PHASE 1112

FIG. 11

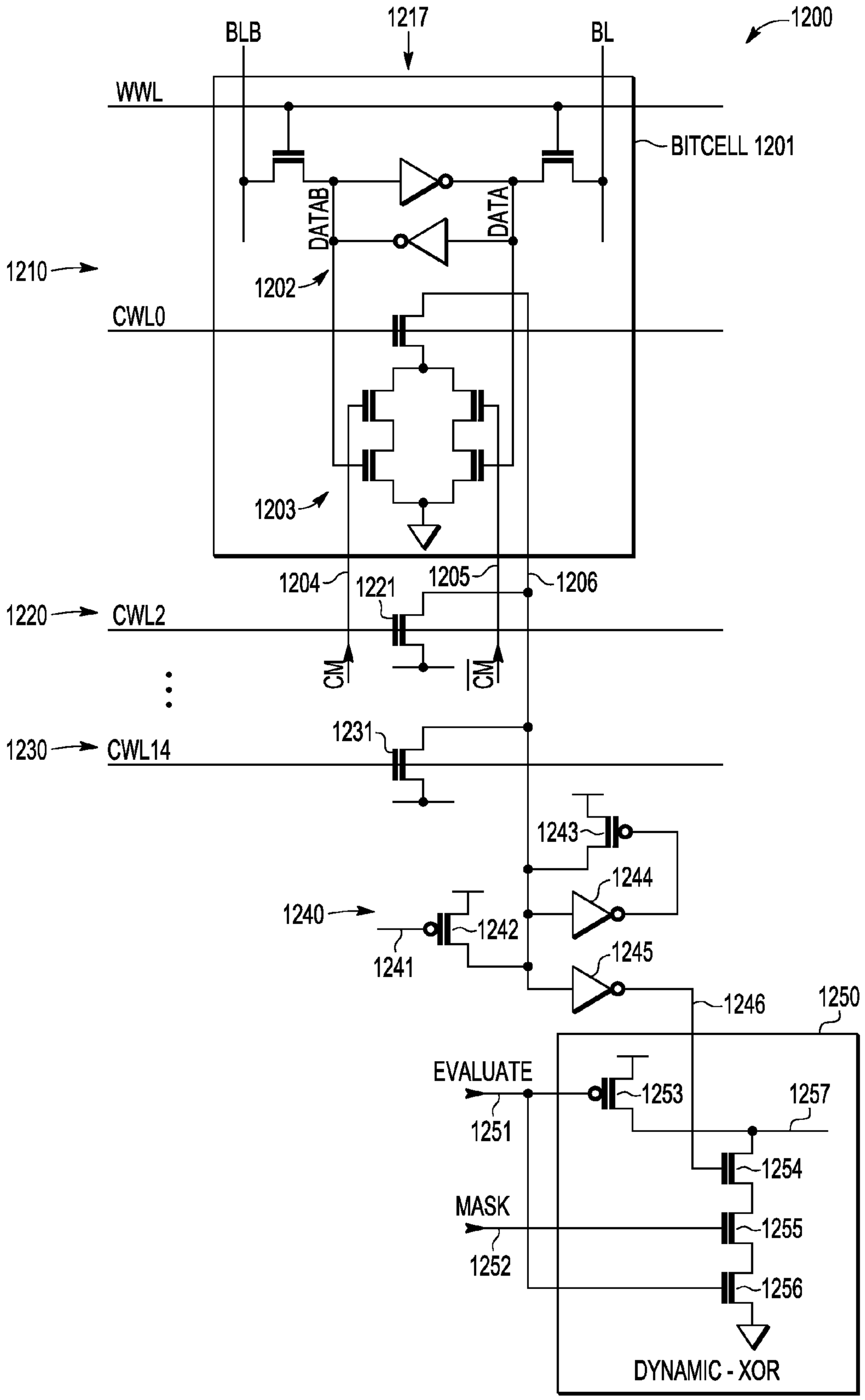

1200
1217
BLB
BL
WWL
BITCELL 1201
DATAB
DATA
1210
1202
CWL0
1203
1204
1221
1205
1206
1220
CWL2
CM
CM
1231
1230
CWL14
1243
1244
1240
1242
1241
1245
1246
1250
EVALUATE
1253
1257
1251
1254
MASK
1255
1252
1256
DYNAMIC - XOR

FIG. 12

MEMORY MANAGEMENT UNIT TAG MEMORY WITH CAM EVALUATE SIGNAL

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is related to the following application which is incorporated by reference: application Ser. No. 13/213,900, filed Aug. 19, 2011, titled "Memory Management Unit TAG Memory."

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention is directed in general to the field of electronic circuits. In one aspect, the present invention relates to a memory access apparatus and method for accessing memory by generating way hit/miss information from base and offset address components.

Description of the Related Art

Data is stored or read from memory at a memory address that is typically computed by adding a base address to an offset address in order to arrive at an effective address for the data. For example, base+offset addressing is used to address memory within data caches, instruction caches, and table-lookaside-buffers (TLBs) as well as data or instructions within other CPU memory units. With such memories, the base and offset values of the memory address are used to determine if the addressed information is stored in the memory. Typically a TAG memory makes this determination by storing addresses of stored information and comparing a TAG portion of the address to the stored addresses to determine if the stored information is present in the memory. A determination that the stored information is present is typically called a hit, and the processing required to make this determination is generally time-consuming because the memory storing the TAG portions of the addresses must be accessed and then compared. In addition, the addition of the base and offset values typically performed to arrive at the effective address usually takes at least two cycles to access the memory. In the first cycle, the base and offset addresses are added, and in the second cycle, the memory is accessed. Since at least two cycles are used to access the memory in a traditional processor, the cycle immediately following a load instruction cannot use the result of the load operation. This delay is referred to as "load latency." Load latency is a performance limitation factor in traditional processors. Load latency often manifests itself in a pipelined processor as a load-use penalty with the load results being unavailable for two machine cycles.

Accordingly, a need exists for an improved memory access circuit, system and method that addresses various problems in the art that have been discovered by the above-named inventors where various limitations and disadvantages of conventional solutions and technologies will become apparent to one of skill in the art after reviewing the remainder of the present application with reference to the drawings and detailed description which follow, though it should be understood that this description of the related art section is not intended to serve as an admission that the described subject matter is prior art.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention may be understood, and its numerous objects, features and advantages obtained, when the following detailed description is considered in conjunction with the following drawings, in which:

FIG. **1** is a simplified schematic block diagram of a fast L1 memory management unit (MMU) TAG circuit in accordance with selected embodiments of the invention;

FIG. **2** is a timing diagram illustrating the computation of the effective address in parallel with the computation of the TAG and data access and the selection of the hit/miss information to reduce the load-store path delay associated with L1 cache access;

FIG. **3** is a simplified schematic block diagram of a TAG lookup and hit/miss generation circuit which is speculatively accessed by two operands in accordance with of the selected embodiments of the invention;

FIG. **4** is a high level flowchart of the steps for using two operands to speculatively access a TAG circuit to generate hit/miss information;

FIG. **5** is a simplified schematic block diagram of an L1 TAG array which uses fast address decoders and PGZ generation logic to generate speculative miss signals from the even and odd arrays and dynamic XOR logic blocks which are selected by the least significant index bit value in accordance with selected embodiments of the invention;

FIG. **6** is a simplified circuit schematic block diagram of the staged compare operation of the A+B=K indexed CAM bitcells which generate speculative even and odd MISS signals for selection by the last significant bit of an index address in accordance with selected embodiments of the invention;

FIG. **7** is a simplified schematic block diagram of an L1 TAG array with permission bits and tag bits in accordance with selected embodiments of the invention;

FIG. **8** is a circuit schematic diagram of a permission bitcell in accordance with selected embodiments of the invention;

FIG. **9** is a circuit schematic diagram of a tag bitcell with partial A+B=K logic in accordance with selected embodiments of the invention;

FIG. **10** is a circuit schematic diagram depicting TAG bit column logic in an even array where delayed evaluate signals are provided to the dynamic comparator in accordance with selected embodiments of the invention;

FIG. **11** is a circuit schematic diagram depicting a bitcell comparator circuit wherein a delayed evaluate signal is used to compare the carry-in value with the carry-out value in accordance with selected embodiments of the invention;

FIG. **12** is a circuit schematic diagram depicting permission bit column logic in an even array in accordance with selected embodiments of the invention.

DETAILED DESCRIPTION

A memory array access circuit is disclosed having a TAG circuit placed in parallel with an operand adder to allow first and second operands to directly access the TAG circuit and generate the way miss/hit signal using a dynamic comparator with a delayed evaluate signal without using the output sum of the operand adder which adds the first and second operands. With the parallel connection of the TAG and operand adder circuits, the operands (e.g., base and offset operands) are applied directly to the TAG circuit using an index CAM bitcell array to generate speculative way hit/miss information by first performing two speculative compares in two rows (one even, and one odd) and then generating two speculative hit/miss signals (even miss, and odd miss) which are selected under control of the least significant index bit computed from the first and second

operands. To access rows in the CAM bitcell array, selected index bits from the first and second operands are encoded with propagate, generate, zero (PGZ) logic to generate a set of PGZ values which are used to select a plurality of speculative indexed CAM wordlines. At the same time, selected tag bits from the first and second operands are encoded with PG logic to generate a set of PG values. In parallel with generating the PG(Z) values, a carry value for the least significant bit of the selected tag bits is computed from the first and second operands by adding other bits in the first and second operands. In addition, a sum value corresponding to the least significant index bit is computed from selected bits in the first and second operands. The set of PG values and the computed carry value are applied to generate a plurality of speculative hit/miss signals by designing each index CAM bitcell in the array to include embedded partial A+B=K compare logic so that the application of the set of PG values and the computed carry value causes the selected CAM bitcells to perform both speculative indexed CAM look up (modified Read Word Line (RWL)) and a partial A+B=K compare operation, resulting in a plurality of possible memory array entry addresses. The computed sum value determines which of the plurality of speculative hit/miss signals is selected from the memory array.

Various illustrative embodiments of the present invention will now be described in detail with reference to the accompanying figures which illustrate functional and/or logical block components and various processing steps. It should be appreciated that such block components may be realized by any number of hardware, software, and/or firmware components configured to perform the specified functions. While various details are set forth in the following description, it will be appreciated that the present invention may be practiced without these specific details, and that numerous implementation-specific decisions may be made to the invention described herein to achieve the device designer's specific goals, such as compliance with process technology or design-related constraints, which will vary from one implementation to another. While such a development effort might be complex and time-consuming, it would nevertheless be a routine undertaking for those of ordinary skill in the art having the benefit of this disclosure. For example, selected embodiments of the present invention are implemented, for the most part, with electronic components and circuits known to those skilled in the art, and as a result, circuit details have not been explained in any greater extent since such details are well known and not considered necessary to teach one skilled in the art of how to make or use the present invention. In addition, selected aspects are depicted with reference to simplified circuit schematics, logic diagrams, and flow chart drawings without including every circuit detail or feature in order to avoid limiting or obscuring the present invention. Such descriptions and representations are used by those skilled in the art to describe and convey the substance of their work to others skilled in the art.

Turning now to FIG. **1**, there is depicted a simplified schematic block diagram of a multi-way L1 MMU TAG circuit **100** in accordance with selected embodiments of the invention. The depicted multi-way TAG circuit **100** receives two multi-bit operands **111**, **113** (OP_A and OP_B) and generates therefrom a hit/miss signal from each way by placing the L1 MMU TAG array(s) **114-117** in parallel to the operand adder circuit **112** in the pipeline stage. In operation, the hit/miss signal for each way is generated by accessing a speculative indexed CAM array **114-117** for each way with selected bits from the operands **111**, **113** and then performing an index CAM operation using selected bits from the operands **111**, **113** to generate speculative hit/miss information from each way. In support of this operation, the multi-way L1 MMU TAG circuit **100** includes an operand adder circuit **112** which is configured to add some or all of the bits (e.g., bits **0**:**63**) from the two multi-bit operands **111**, **113** (OP_A and OP_B), including any required carry values (e.g., the carry out value from bit position **48**) and sum values (e.g., the sum value from bit position **51**) for one or more bit positions in the resulting sum. To avoid delay associated with waiting for the operand adder circuit **112** to compute the sum of the multi-bit operands **111**, **113** (OP_A and OP_B), selected operand bits are applied directly to the indexed CAM array for each way to generate speculative hit/miss information. For example, selected index bits (e.g., bits **48**:**51**) from each operand **111**, **113** are applied to a first speculative indexed CAM array **114** for way **0** to generate speculative indexed CAM wordlines from way **0**, while selected tag bits (e.g., bits **0**:**47**) from each operand **111**, **113** and a computed carry-in value **125** for the least significant tag bit (e.g., $Cout_{48}$) are applied to the first speculative indexed CAM array **114** for way **0** to generate two (even and odd) speculative hit/miss signals **118**, **119** from way **0**. Under control of the computed sum value **124** for the least significant index bit, the selection or multiplexer circuit **120** selects the speculative hit/miss signal **126** for way **0**.

Speculative hit/miss signal information may also be generated from other ways in accordance the multi-way embodiments disclosed herein. For example, selected index bits (e.g., bits **48**:**51**) from each operand **111**, **113** may also be applied to a second speculative indexed CAM array **115** to generate speculative indexed CAM wordlines from way **1**, while selected tag bits (e.g., bits **0**:**47**) from each operand **111**, **113** and a computed carry-in value **125** for the least significant tag bit (e.g., $Cout_{48}$) are applied to the second speculative indexed CAM array **115** to generate two (even and odd) speculative hit/miss signals from way **1** which are selected under control of the computed sum value **124** by the selection/multiplexer circuit **121** for output as the speculative hit/miss signal **126** for way **1**. Likewise, additional dedicated CAM arrays **116-117** and selection/multiplexer circuits **122-123** may be provided for each additional way.

With the disclosed L1 MMU TAG circuit, memory access speeds are improved over conventional memory address schemes which use two or more cycles to compute an effective address by adding a base and offset address (using a first cycle) and then use the effective address to access the memory with the effective address (using a second cycle). In contrast, the L1 MMU TAG lookup and generation of way hit/miss information as disclosed herein is faster since, in the L1 MMU TAG circuit, the operands are used to directly access the tag array to generate hit/miss information, thereby reducing load latency in the load-store timing path. The reduced latency is depicted in FIG. **2** which illustrates a timing diagram **200** showing how the computation of the effective address (EA operation **220**) in parallel with the computation of the TAG and data access (tag access operation **240**) and the selection of the hit/miss information ("=" operation **260**) all occur within a phase of a cycle (t_{cycle}) of the clock **210**, thereby reducing the load-store path delay associated with conventional L1 cache access which can require two or more clock cycles.

FIG. **3** is a simplified schematic block diagram of a data processing system **300** in which a TAG lookup and hit/miss generation circuit **310** which is speculatively accessed by two operands (OP_A and OP_B) to perform a staged comparison with speculative indexed CAM arrays to generate a

speculative miss signal in accordance with selected embodiments of the invention. In selected embodiments, one of the operands (Operand A) provides the "base" address and the other operand (Operand B) provides the "offset" address such that the base and offset addresses are used to generate the "effective" address of the memory entry. However, instead of using the effective address to generate a miss signal, selected index bits in the operands OP_A, OP_B are used to speculatively access a plurality of indexed CAM wordlines. In selected embodiments where index bits **48** through **51** are used, bit **48** may be the most-significant-bit (MSB) and bit **51** may be the least-significant bit (LSB) of the index bits, though the significance of the bits might be reversed so that the higher-numbered bit is more significant than the lower-numbered bit.

In any event, the selected index bits **306**, **303** from operands OP_A and OP_B may be provided as memory address bits to the PGZ generation logic **311** which combines pairs of bits using logical operators (e.g., NOT, XOR, OR, AND, NAND) to create propagate, generate, zero (PGZ) logic values **314**, **315** that are provided to the L1 MMU tag array **313**. In the example shown, four bits ($A_{(48:51)}$, $B_{(48:51)}$) are provided to the PGZ generation logic **311** from each of the first and second operands OP_A, OP_B to generate PGZ values in the form of two speculative addresses **314**, **315**. Based on the index bits **306**, **303** from operands OP_A and OP_B, the PGZ generation logic **311** generates PGZ values from each index bit pair which may include a Propagate value (by XORing the input bit pair), a Generate value (by ANDing the input bit pair), and a Zero value (by ANDing the inverted input bit pair). In addition, complementary $\overline{P}$ and $\overline{G}$ values may be generated, with $\overline{P}$ being the inverse of the XOR value (by XNORing the input bit pair), and with $\overline{G}$ being the inverse of the AND value (by NANDing the input bit pair). As will be appreciated, the logical operations may be performed with equivalent circuitry and are not necessarily performed with AND and XOR logic gates.

The PGZ values **314**, **315**, in turn, are provided to copies of wordline generators **322**, **332** in order to access a memory entry from the odd and even memory arrays. As a result of running the PGZ values **314**, **315** through the wordline generators **322**, **332**, respectively, one of the odd wordlines **323***a-g* will be enabled and one of the even wordlines **333***a-g* will be enabled. In this way, the PGZ values **314** are processed by a plurality of odd wordline generators **322** to generate one possible odd memory array address in the odd array **323***a-g*, while the PGZ values **315** are processed by a plurality of even wordline generators **332** to generate one possible even memory array address in the even array **333***a-g*.

In parallel with the PGZ generation logic **311**, selected tag bits **305**, **302** from operands OP_A and OP_B are provided as memory address bits to the PG generation logic **312** which generates PG logic values **316**, **317** from the tag bits **0** through **47**. In the example shown, tag bits $A_{(0:47)}$, $B_{(0:47)}$ are provided to the PG generation logic **312** from each of the first and second operands OP_A, OP_B to generate PG values **316**, **317** from each tag bit pair which may include a Propagate value (by XORing the input bit pair), a Generate value (by ANDing the input bit pair), and complementary $\overline{P}$ value (by XNORing the input bit pair). In addition, fast carry generation logic **301** processes bits **48** through **63** from operands OP_A and OP_B to compute a carry value for the least significant bit of the tag bits (e.g., carry out Cout (**48**) from bit position **48**). By providing the PG values **316**, **317** to index CAM bitcells in the odd and even arrays **323**, **333**, two speculative miss (even miss and odd miss) signals **318**, **319** are generated through the dynamic XOR blocks **340**, **341** for the even and odd arrays under the control of the computed carry value Cout (**48**).

In parallel with memory array access operations through the PG(Z) values **314-317**, fast carry generation and sum logic **351** processes selected bits **304**, **307** (e.g., bits **52** through **63**) from operands OP_A and OP_B to compute a carry value for the least significant bit of the index bits (e.g., carry out Cout (**52**) from bit position **52**). In addition, the carry value is added to the LSB of the memory address bits of the Operands A and B. This results in sum value **352** (and possibly its complement) for the LSB of the memory address bits (e.g., least significant index bit **51**). The sum value **352** is applied to the match selector and latch **353** to control the selection of either the possible odd memory array entry address **318** or the possible even memory array entry address **319**, depending on the computed sum value **352**. The selected memory array address **354** is then retrieved from memory array **355**.

Turning now to FIG. **4**, there is illustrated a high level flowchart **400** of the steps for using two operands to speculatively access a TAG circuit to generate hit/miss information. Operand A **401** and Operand B **402** each include a plurality of bits, some of which are used to address a memory entry in a memory array. In an example embodiment, each operand includes 64 bits numbered 0 to 63, where bit **0** is the most significant bit, and bit **63** is the least significant bit. As described herein, selected index bits in the operands (e.g., bits **48** through **51**) are used to speculatively access a plurality of possible memory array entry addresses in even and odd arrays, while selected tag bits in the operands (e.g., bits **0** through **47**) are used with a carry out value for the most significant index bit are used to generate two speculative miss (even miss and odd miss) signals. Again, while bit **0** is the most-significant-bit (MSB) and bit **63** is the least-significant-bit (LSB), the significance of the bits might be reversed so that the higher-numbered bit is more significant than the lower-numbered bit.

At step **403**, Operand A **401** and Operand B **402** are received. In the example shown, Operand A **401** may represent a "base" address and the Operand B **402** may represent an offset address, where the base and offset addresses are used to generate an "effective" address of the memory entry. Instead of adding Operand A **401** and Operand B **402** with an adder circuit, three parallel processes commence at this point. A first process evaluates index address bits (e.g., bits **48** through **51**) to arrive at two possible wordlines (as used herein, a "wordline" is an address of an entry in the memory array or an actual memory array entry, as the context indicates). A second process evaluates tag address bits (e.g., bits **0** through **47**) and a computed carry value for the least significant tag bit to arrive at two possible speculative miss signals (as used herein, a "miss" signal can be considered a "hit" signal, depending on the polarity of the logic and as the context indicates). A third process determines if a carry results from bits in the operands (e.g., bits **52** through **63**) and adds the carry value to the LSBs of the bits of the Operand A and B used to address the memory entry. The summation value determines which of the possible speculative miss signals is the actual miss signal.

In the first parallel process, the memory access index bits for accessing the memory array (e.g., bits **48** through **51** for both Operands A and B) are run through PGZ generation logic at step **404**. PGZ generation logic combines pairs of bits using logical operators (XOR, OR, AND, NAND) to create PGZ values for each of the memory access index bits

from each operand. As indicated by the cascaded blocks at step **404**, PGZ values are generated for each index bit from each operand. Thus, four bits are provided from each operand (Operands A and B) as base and offset addresses to generate a four bit effective address from each operand by generating PGZ values for the MSBs (bit **48** from both operands), bit **49** from both operands, bit **50** from both operands and from the LSBs (bit **51** from both operands). Each effective address generated at step **404** can be used to access a memory entry from a multi-entry memory array at step **409** by running the PGZ access values for each bit through wordline generators. As described with reference to FIG. **3**, multiple copies of the wordline generators may be used, depending on the size of the memory array being accessed. For example, with a sixteen-entry memory array, sixteen copies of the wordline generators are used (eight copies of the odd wordline generator **322** and eight copies of the even wordline generator **332**).

In the second parallel process, tag address bits (e.g., bits **0** through **47**) and a fast carry value for the least significant tag bit are evaluated to arrive at two possible speculative miss signals. The process begins at step **405** by performing a fast carry computation for bits **48** through **63** for both operands A and B, thereby generating the carry out value (Carry_Out) for the most significant memory array access (index) bit (e.g., bit **48**). In addition, the memory array comparison (tag) bits (e.g., bits **0** through **47** for both Operands A and B) are run through PG generation logic at step **407** to generate PG Comparison values (e.g., P, $\overline{P}$ and G). At step **410**, the resulting PG comparison values (generated at step **407**) and the computed carry out value (Carry_Out) are run through the odd and even indexed CAM arrays.

At step **412**, the results of the wordline generators (from step **409**) and the odd and even indexed CAM arrays (from step **410**) are received. The results of running the PG(Z) values through the wordline generators and indexed CAM arrays are one possible even wordline (with 0 being considered an even wordline, i.e., **0**, **2**, **4**, **6**, **8**, **10**, **12**, and **14**) and one possible odd wordline (i.e., **1**, **3**, **5**, **7**, **9**, **11**, **13**, **15**). In the example shown, even wordline **416** and odd wordline **417** have been identified as the possible wordlines within memory array **415**.

The reason that there are two wordline possibilities is that there may be a carry resulting from the operand bits that are less significant than the memory array access (index) bits. To resolve this uncertainty, the third parallel process is undertaken to select between the even and odd wordlines **416**, **417**. As illustrated, the third process begins at step **406** by performing a fast carry computation up to (but not including) the least significant memory array access (index) bit (e.g., for bits **52** through **63**) for both Operands A and B. In step **408**, the carry result value generated in step **406** is summed (added) to the least-significant memory array access (index) bits (e.g., bit **51**) of the Operands A and B. A determination is made at step **411** as to whether the sum operation results in a "1" or a "0". If the sum operation results in a "0" (negative outcome to decision step **411**), the even wordline **416** is selected (step **413**). On the other hand, if the sum operation results in a "1" (affirmative outcome to decision step **411**), the odd possible wordline **417** is selected (step **414**). At step **418**, the selected wordline is retrieved from memory array **415**.

Turning now to FIG. **5**, there is depicted a simplified schematic block diagram of an L1 TAG array **500** which uses fast address decoders **501**, **502** and PGZ generation logic **505** to generate speculative miss signals from the even and odd arrays **503**, **508** and dynamic XOR logic blocks **504**, **507** which are selected by the least significant index bit value. As depicted, selected bits (e.g., index bits **48**:**51**) from each operand are used for speculative access of the even and odd rows in the even array **503** and odd array **508**, respectively. For example, if the L1 TAG array **500** has sixteen memory array entries, the rows in the even array **503** store eight even K=A+B values (e.g., A+B=0, A+B=2, . . . A+B=14), while the rows in the odd array **508** store eight odd K=A+B values (e.g., A+B=1, A+B=3, . . . A+B=15). Using the selected index bits (e.g., bits **48**:**51**) from Operands A and B, the fast address decoders **501**, **502** speculatively access the rows in the even and odd arrays **503**, **508**.

At the same time, selected tag bits (e.g., bits **0**:**47**) from Operands A and B are applied to the PG generation logic **505** to generate PG comparison values. As will be appreciated, the PG generation logic **505** may be shared across multiple ways (e.g., 2 or 4), and is used to generate a propagate bit ($P_i=A_i \otimes B_i$) and its complement $\overline{P}_i$, a generate bit ($G_i=A_i \cdot B_i$). The PG comparison values (e.g., P, $\overline{P}$ and G) for each bit position are applied to A+B=K indexed CAM cells in the even and odd arrays **503**, **508** to generate required carry in and produced carry out values, thereby implementing a two-stage comparison of the operand bit values A_i, B_i and the stored K_i values in the array. In the first comparison stage, the PG comparison values generated by the PG generation logic **505** are applied to A+B=K indexed CAM cells from the speculatively accessed even and odd rows in the even and odd arrays **503**, **508**. At this first stage, the required carry in and produced carry out values are generated by the A+B=K indexed CAM cells. In the second comparison stage, the required carry in and produced carry out values output from the A+B=K indexed CAM cells in the even and odd arrays **503**, **508** are applied to the dynamic XOR logic **504**, **507**. As explained more fully below, the dynamic XOR logic **504**, **507** uses a delayed evaluate signal to effectively perform a speculative comparison on the two speculatively accessed rows to generate two speculative miss signals (output=1) or speculative hit signals (output=0), depending on the polarity and logic used. Using additional combinatorial logic circuitry (not shown), the speculative even and odd hit/miss signals may be logically combined with a sum value for the least significant index bit to select between the speculative even and odd hit/miss signals.

For additional detail, reference is now made to FIG. **6** which depicts a simplified circuit schematic block diagram **600** of the staged compare operation of the A+B=K indexed CAM bitcells to generate speculative even and odd MISS signals for selection by the last significant bit of an index address. As depicted, selected bits (e.g., bits **0**:**47**) from Operands A and B are applied to the PG generation logic **605** to generate PG comparison values **606** for each of the selected bits for purposes of implementing a K=A+B comparator. In a first comparison stage **610**, the PG comparison values **606** are applied to the even and odd indexed CAM bitcells **603**, **608** which each include logic circuitry for performing a partial A+B=K comparison operation to generate required carry in and produced carry out values **613**, **618**. In the second comparison stage **620**, the required carry in and produced carry out values **613**, **618** from the even and odd arrays are applied to the corresponding dynamic XOR logic **604**, **607** which effectively performs two speculative compares on the two speculatively accessed rows to generate two speculative miss signals **614**, **617**, respectively.

To illustrate selected embodiments in which indexed CAM cells are arranged in rows with associated permission bitcells, FIG. **7** depicts a simplified schematic block diagram

of an L1 TAG array **700** with permission bits and tag bits arranged in rows and columns in accordance with selected embodiments of the invention. The depicted array **700** stores "even" K values (e.g., A+B=0, A+B=2, . . . A+B=8), while another array (not shown) stores "odd" K values. In particular, the array **700** includes a first even row (e.g., Row **0**) in which a plurality of permission bits **701-702** and tag bits **703-709** share a common write wordline (WWL**0**) and compare wordline (CWL**0**). Additional even rows are included, such as a last even row (e.g., Row **8**) in which a plurality of permission bits **711-712** and tag bits **713-719** share a common write wordline (WWL**8**) and compare wordline (CWL**8**).

As described herein, each of the tag bits (e.g., **703-709**, **713-719**) may be implemented as an indexed CAM bitcell which includes logic circuitry for performing a partial A+B=K comparison operation in response to computed PG comparison values. Thus, "even" tag bit **703** receives PG comparison values **731** and/or complementary PG comparison values **732** from PG generation block **730**. Upon application of the PG comparison values **731**, **732**, the logic circuitry in the tag bit **703** generates one or more carry values **723** which are provided to the dynamic XOR circuit block **720**. In similar fashion, each of the remaining tag bits **704-709** in the first row receives a corresponding pair of PG comparison values (e.g., **733/734**, **735/736**, **737/738**) and generates corresponding carry values **724-726** which are provided to the dynamic XOR circuit block **720** (as indicated), and the same design and operation are used for the other tag bits in other rows (e.g., tag bits **713-719**).

The permission bits **701-702**, **711-712** for each row are used as "valid" or "status" bits to control the hit/miss signal generated by a given row. Because the permission bits are known beforehand, there is no need to perform speculative comparison with operand bits, so PG comparison values are not provided to the permission bits by the PG generation block **730**. The permission bits (e.g., **701**, **702**) associated with a row (e.g., Row **0**) each operate to provide a "valid" bit function so that, if the permission bit is set to "zero" (indicating "not valid), there will be a MISS produced even if there is a match otherwise indicated by the memory address portion (bits **0**:**51**) from Operands A and B.

An example embodiment of the permission bitcell design is shown with the circuit schematic diagram shown in FIG. **8**. As depicted, the depicted permission bitcell **800** includes a memory unit **801** of two inverters **802**, **803** cross-coupled between first and second data nodes (DATA and DATAB). The first and second data nodes are connected across access transistors **820**, **810** to a shared bit line pair BL, BLB. Write access to the permission bitcell **800** is controlled by a write wordline (WWL) that is applied to the gates of the access transistors **810**, **820**. In addition, a comparison circuit is connected between the data nodes for generating a permission output signal **883** in response to first and second control signals (CM, $\overline{CM}$) and a compare wordline (CWL) signal for the permission bitcell (row). The comparison circuit includes a first pair of series-connected transistors **840**, **850** coupled in parallel with a second pair of series-connected transistors **860**, **870** between a connection transistor **830** and a reference supply voltage (e.g., ground). As depicted, the first and second data nodes (DATA and DATAB) are connected, respectively, to the gates of the transistors **870**, **850**, and the first and second control signals (CM, $\overline{CM}$) are connected, respectively, to the gates of the transistors **840**, **860**. In addition, the connection transistor **830** connects the pair of series-connected transistors **840**, **850**, **860**, **870** to the permission output signal **883**, with the compare wordline (CWL) signal connected to the gate of the connection transistor **830**. In this example, a logical "zero" stored in the data node DATA will generate a logical "zero" as the permission output signal **883** when the first and second control signals are set at CM=1 and $\overline{CM}$=0.

Turning now to FIG. **9**, there is depicted a circuit schematic diagram of a tag bitcell **900** with embedded partial A+B=K logic in accordance with selected embodiments of the invention. As depicted, the depicted tag bitcell **900** includes a memory unit **901** of two inverters **902**, **903** cross-coupled between first and second data nodes (DATA and DATAB). The first and second data nodes are connected across access transistors **910**, **920** to a shared bit line pair BL, BLB. Write access to the tag bitcell **900** is controlled by a write wordline (WWL) that is applied to the gates of the access transistors **910**, **920**. In addition, embedded partial A+B=K comparison logic circuitry is connected to the data nodes for generating carry values **953**, **957** in response to PG comparison values (P, $\overline{P}$, G) and a compare wordline (CWL) signal for the tag bitcell (row).

As depicted, the embedded comparison logic circuitry includes a first logic circuit **930** in which a first transistor **934** is coupled in parallel with a pair of series-connected transistors **932**, **933** between a connection transistor **931** and a reference supply voltage (e.g., ground). As depicted, a Generate (G) value **952** generated by the PG generation logic is connected to the gate of the first transistor **934**, a Propagate (P) value **954** generated by the PG generation logic is connected to the gate of the transistor **932**, and the second data node DATAB is connected to the gate of the transistor **933**. In addition, the connection transistor **931** connects the first logic circuit **930** to output the carry-out value C_{out} **953**, with the compare wordline (CWL) signal connected to the gate of the connection transistor **931**.

The embedded comparison logic circuitry also includes a second logic circuit **940** in which a first pair of series-connected transistors **942**, **943** is coupled in parallel with a second pair of series-connected transistors **944**, **945** between a connection transistor **941** and a reference supply voltage (e.g., ground). As depicted, the Propagate (P) value **954** generated by the PG generation logic is connected to the gate of the transistor **942**, the second data node DATAB is connected to the gate of the transistor **943**, the complementary Propagate ($\overline{P}$) value **955** generated by the PGZ generation logic is connected to the gate of the transistor **944**, and the first data node DATA is connected to the gate of the transistor **945**. In addition, the connection transistor **941** connects the second logic circuit **940** to output the carry-in value C_{in} **956**, with the compare wordline (CWL) signal connected to the gate of the connection transistor **941**.

Using only the first and second logic circuits (e.g., **930**, **940**) for embedded comparison logic circuitry in each bitcell to generate the C_{out} value **953** and C_{in} value **956**, the resulting CAM array can be built much more compactly than with bitcells which include additional comparison logic circuitry. In addition, the layout requirements for delivering PG comparison values are greatly reduced as compared to embedded comparison logic circuitry which requires additional PGZ values (e.g., Z, $\overline{Z}$, G, $\overline{G}$, P, and $\overline{P}$) in order to generate the additional complementary versions of the C_{out} and C_{in} values. However, the reduced area and improved layout requirements are achieved at the expense of generating only the carry values C_{in} and C_{out}, and not their complements. As explained below, this effectively imposes a delay in completing the A+B=K logic operation in order to compute the complementary carry values.

Using the embedded comparison logic circuitry, the tag bitcell **900** can be used to determine if the address Operands A and B correspond to the stored K value by evaluating the "A+B=K" conditions without carry propagation that would be required by adding the Operands A and B. This evaluation requires only that, knowing A and B, one also knows what the carry into each bit must be if K=A+B. With this approach, only the adjacent pairs of bits need to be checked to verify that the previous bit produces the carry required by the current bit, and then a "ones" detector can be used to check that the condition is true for all N pairs. Specifically, if K=A+B for bitcell i, the required carry-in value (C_{in}, Required=$A_i \otimes B_i \otimes K_i$) must match the produced carry-out value from bitcell i−1 (C_{outi-1} Produced=$(A_{-1} \otimes B_{i-1}) \cdot \overline{K}_{i-1} + A_{i-1} \cdot B_{i-1}$). Conversely, if carry-in value (C_{ini} Required) does not match the produced carry-out value (C_{outi-1} Produced), then a miss is indicated. The bitcell i−1 is the adjacent less significant bit of bitcell i.

To support the embedded comparison logic circuitry, the tag bitcell **900** receives PG comparison values from the PG generation logic. In the example embodiment shown in FIG. **9**, the tag bitcell **900** receives a Generate bit **952** ($G_i=A_i \cdot B_i$), a Propagate bit **954** ($P_i=A_i \otimes B_i$), and a Not Propagate bit **955** ($\overline{P}_i=A_i\ B_i$). In this way, the carry values generated by the tag bitcell **900** include a first required carry-in value **956** (e.g., C_{ini} Required=$A_i \otimes B_i \otimes K_i$) and a first produced carry-out value **953** (e.g., C_{outi} Produced=$(A_i \otimes B_i) \cdot \overline{K}_i + A_i \cdot B_i$). In other embodiments, additional embedded comparison logic circuitry could be included to generate complementary versions of the required carry in value ($\overline{C_{ini}}$) and produced carry out value ($\overline{C_{outi}}$). The generated carry values **953**, **956** from tag bitcell **900** are provided to the dynamic XOR circuit block which determines if there is a match or HIT between the required carry-in value for the tag bit (e.g., **900**) (e.g., C_{ini} Required) and the first produced carry-out value for the adjacent, less significant tag bit (e.g., C_{outi-1}). Stated another way, the dynamic XOR circuit block determines if there is a MISS when the required carry-in value for the tag bit (e.g., **900**) (e.g., C_{ini} Required) does not match the first produced carry-out value for the adjacent, less significant tag bit (e.g., C_{outi-1}).

To illustrate how the tag bitcells are arrayed and processed to identify a speculative miss signals for selection by the least significant bit of the index address bits, reference is now made to FIG. **10** which depicts a circuit schematic diagram of the TAG bit column logic **1000** in an even array **1019** in accordance with selected embodiments of the invention. The depicted array **1019** includes a plurality of tag bitcells (e.g., **1001**, **1002**) arranged in a plurality of rows **1011**, **1012**, **1013** and columns **1014**, **1015**. Each bitcell (e.g., **1001**, **1002**) may use substantially the same design and operation as shown in FIG. **9** so to include embedded partial A+B=K comparison logic circuitry for generating carry values (e.g., **1004**, **1005**, **1008**, **1009**) in response to PG comparison values (e.g., **1016**, **1017**) and a compare wordline (CWL) signal for the tag bitcell. For example, bitcell **1001** receives PG comparison values (G, P, and $\overline{P}$) at input lines **1016**, and outputs C_{out} at line **1004**, and C_{in} at line **1005**. Similarly, bitcell **1002** receives PG comparison values (G, P, and $\overline{P}$) at input lines **1017**, and outputs C_{out} at line **1008**, and C_{in} at line **1009**.

The generated carry values (e.g., **1004/1005**, **1008/1009**) are applied to the bitline driver output circuit block **1020** which is controlled by the precharge signal line **1021**. Each carry value line is coupled to a keeper pull-up circuit for driving logical "one" values to a predetermined reference voltage (e.g., Vdd) by coupling the carry value line across a first pull-up transistor (e.g., PMOS transistor) to the predetermined reference voltage, where the first pull-up transistor is gated by an inverter which is coupled between the carry value line and the first pull-up transistor. In addition, each carry value line is coupled across a second pull-up transistor (e.g., PMOS transistor) to a predetermined reference voltage (e.g., Vdd) when the precharge signal line **1021** has a first logical state (e.g., "zero" volts). However, when the precharge signal line **1021** has a second logical state (e.g., "one" volt), the second pull-up transistor is not conductive, and the carry values **1004/1005**, **1008/1009** are each applied to a corresponding inverter circuit for purposes of generating inverted carry values **1022/1023**, **1024/1025**. Thus, carry values **1004/1005** from bitcell **1001** become inverted carry values **1022/1023**. Likewise, carry values **1008/1009** from bitcell **1002** become inverted carry values **1024/1025**.

In order to generate carry values and complementary carry values from each of the inverted carry values, each inverted carry value is applied to a pair of series-connected inverters so that the first inverter generates a carry value from the inverted carry value, and so that the second inverter generates a complementary carry value from the carry value. Thus, the inverted carry-out value $\overline{C_{out}}$ **1022** is applied to a first inverter **1026** to generate the carry-out value C_{out} at line **1030**. The output from the first inverter **1026** is applied to a second inverter **1033** to generate the complementary carry-out value $\overline{C_{out}}$ at line **1037**. In similar fashion, the inverted carry-in value $\overline{C_{in}}$ **1023** is applied to the inverter pair **1027**, **1034** to generate the carry-in value C_{in} at line **1031** and the complementary carry-in value $\overline{C_{in}}$ at line **1038**. Likewise, the inverted carry-out value $\overline{C_{out}}$ **1024** is applied to the inverter pair **1028**, **1035** to generate the carry-out value C_{out} at line **1039** and the complementary carry-out value $\overline{C_{out}}$ at line **1040**, while the inverted carry-in value $\overline{C_{in}}$ **1025** is applied to the inverter pair **1029**, **1036** to generate the carry-in value C_{in} at line **1032** and the complementary carry-in value $\overline{C_{in}}$ at line **1041**, and so on. Thus, the inverter structures (e.g., **1027**, **1034**, **1028**, **1035**) are used to locally generate true and complementary carry-in and carry-out signals off the bit lines (e.g., **1023**, **1024**).

The carry values (C_{in} and C_{out}) and their corresponding complementary carry values ($\overline{C_{in}}$ and $\overline{C_{out}}$) from each bitcell (e.g., **1038-1039**, **1040**, **1031**) are applied to the dynamic XOR logic circuit block **1050** which is controlled by an evaluate clock signal **1051** to generate a speculative miss signal **1071** from the even array **1019** if any of the stored K values in the tag bitcells from the accessed row do not match the corresponding Operand A and B values. To accommodate any delays imposed by generating the carry values and their corresponding complementary carry values, delay logic circuitry **1085** may optionally be provided to generate a delayed evaluate clock signal **1051** by logically ORing the compare wordline (CWL) signals from the array **1019** at the dynamic OR gate **1086** so that any compare wordline signal generates a delayed evaluate clock signal **1051** that is provided to the dynamic XOR logic circuit block **1050**.

In operation, the dynamic XOR logic circuit block **1050** dynamically compares the required carry-in value for each tag bitcell in a selected row with the produced carry-out value from the adjacent, less significant tag bitcell. If any of the tag bitcells in a row from the even array **1019** have a required carry-in value that does not match the produced carry-out value from the adjacent, less significant tag bitcell, the dynamic XOR logic circuit block **1050** generates a speculative even miss signal **1071** having a first logical state (e.g., logical "one"). Similarly, if any of the tag bitcells in a row from the odd array (not shown) have a required carry-in

value that does not match the produced carry-out value from the adjacent, less significant tag bitcell, the dynamic XOR logic circuit block for the odd array (not shown) generates a speculative odd miss signal **1072** having a first logical state (e.g., logical "one").

To dynamically compare the carry values, the dynamic XOR logic circuit block **1050** includes a plurality of match circuits **1052**, **1053**, **1054**, etc. Each match circuit may be implemented as an XNOR logical function at the output dynamic node **1055** based on inputs from the required carry-in value for a given tag bitcell with the produced carry-out value from the adjacent, less significant tag bitcell. For example, the dynamic XOR logic circuit block **1050** includes a match circuit **1053** which includes a first pair of series-connected transistors M**1**, M**2** coupled in parallel with a second pair of series-connected transistors M**3**, M**4** between the dynamic node **1055** and a reference voltage connection transistor M**5**. As depicted, the C_{out} value at line **1039** from bitcell **1002** (by virtue of double inversion from the bitline driver output circuit block **1020** and inverter **1028**) is connected to the gate of the first transistor M**1**, the $\overline{C_{in}}$ value from bitcell **1001** (by virtue of triple inversion at the bitline driver output circuit block **1020** and inverter pair **1027**, **1034**) is connected to the gate of the second transistor M**2**, the $\overline{C_{out}}$ value from bitcell **1002** (by virtue of triple inversion at the bitline driver output circuit block **1020** and inverter pair **1028**, **1035**) is connected to the gate of the third transistor M**3**, and the C_{in} value from bitcell **1001** (by virtue of double inversion at the bitline driver output circuit block **1020** and inverter **1027**) is connected to the gate of the fourth transistor M**4**. In addition, the reference voltage connection transistor M**5** connects the pair of series-connected transistors M**1**-M**4** to the predetermined reference voltage (e.g., ground), with the evaluate clock signal **1051** connected to the gate of the reference voltage connection transistor M**5**. In this example, a logical "zero" will be generated at the dynamic node **1055** by the match circuit **1053** if the C_{out} line **1039** (from bitcell **1002**) and the C_{in} line **1031** (from bitcell **1001**) do not match. However, if there is a match, the match circuit **1053** will output a logical "one" at the dynamic node **1055**.

To illustrate the operation of the bitcell comparator circuit, reference is now made to FIG. **11** which depicts a circuit schematic diagram **1100** of the match circuit **1053** (shown in FIG. **10**) wherein a delayed evaluate signal is used to compare the carry-in value from bitcell **1002** with the carry-out value from bitcell **1001** in accordance with selected embodiments of the invention. In an initial precharge phase **1111**, all signal lines (e.g., **1005**, **1008**) for the carry values are pre-charged "high" by virtue of the second pull-up transistor in the bitline driver output circuit block **1020**. As the precharged "high" signal propagates through the inverters **1101**, **1102** (from the bitline driver output circuit block **1020**) and the inverter pairs **1027**/**1034**, **1028**/**1035**, precharge gate voltage values b=0, $\overline{a}$=1, a=0, $\overline{b}$=1 are applied to the gates of the transistors M**1**, M**2**, M**3**, and M**4**, respectively. In this precharge phase **1111**, the match circuit **1053** will not provide a conductive path between the dynamic node **1055** and the reference voltage connection transistor M**5**. In a comparison phase **1112**, the match circuit **1053** propagates the carry-in value from bitcell **1002** (appearing on signal line **1008**) and the carry-out value from bitcell **1001** (appearing on signal line **1005**). Obviously, if the carry-in and carry-out values are the same as the precharge values (e.g., C_{ini}(H)=C_{outi-1}(H)="high"), then nothing changes at the gates M**1**-M**4** of the matching circuit **1053** and there is no conductive path established between the dynamic node **1055** and the reference voltage connection transistor M**5**. Even if the carry-in and carry-out values are both set to a "low" value, the applied gate voltage values change to b=1, $\overline{a}$=0, a=1, $\overline{b}$=0 after a transition phase, and there is still no conductive path established by the matching circuit **1053** during the comparison phase **1112**. However, if the carry-in and carry-out values are not the same (e.g., C_{ini}=high, but C_{outi-1}="low"), then gate voltage values change to b=0, $\overline{a}$=0, a=1, $\overline{b}$=1 so that the gates M**1**-M**4** of the matching circuit **1053** establish a conductive path between the dynamic node **1055** and the reference voltage connection transistor M**5**, which in turn pulls the dynamic node **1055** "low" to the reference voltage in response to the delayed evaluate signal **1051**. As depicted by the timing requirements for transitions in the gate voltage values b, $\overline{a}$, a, $\overline{b}$, there is a transition phase between the precharge phase **1111** and comparison phase **1112** while signal are propagating through the matching circuit **1053** which is factored into the delay required for the delayed evaluate clock signal **1051**.

As depicted, the plurality of XNOR match circuits **1052**, **1053**, **1054** are connected in parallel to the dynamic node **1055** which in turn is connected to a first pull-up circuit **1056**. By connecting the dynamic node **1055** as a first input to the NAND gate circuit **1059** which has its other input **1058** held to a logical "one" by the pull-up circuit **1057**, the resulting output **1071** from the NAND gate **1059** will signal a speculative miss (e.g., logical value "zero") from the even array **1019** if any of the match circuits **1052**, **1053**, **1054** indicate that C_{outi-1} does not match C_{ini}. Conversely, the output **1071** from the NAND gate **1059** will signal a speculative hit (e.g., logical value "one") from the even array **1019** only if all of the match circuits **1052**, **1053**, **1054** on a selected row indicate that C_{outi-1} matches C_{ini}.

While the dynamic XOR logic circuit block **1050** dynamically compares the produced C_{out} from the adjacent less significant bit and required C_{in} values for each bit positions in **0**:**46**, a static XOR logic circuit **1060** may be used to generate an XOR output **1062** between the produced C_{out} value of bit **48** **1061** and the required C_{in} value for LSB **47**. In an example embodiment shown in FIG. **10**, the static XOR logic circuit **1060** may be implemented with the depicted PMOS and NMOS transistor stack which performs the XOR operation between the required C_{in}, $\overline{C_{in}}$values for LSB **47** and the produced C_{out} value of bit **48** ($Cout_{48}$) on line **1061** to check if they match or not. The static XOR logic circuit **1060** can be used in place of a dynamic XOR gate operation when the C_{out} value of bit **48** provided from external logic is static. The match signal **1062** from the static XOR compare operation is effectively "OR"ed with the dynamic compare signal **1071** using the complex OR-AND-Invert (OAI) gate **1064** which performs an OR-AND-INVERT function. Insofar as the same comparison is used for the odd array, FIG. **10** shows that the match signal **1062** from the static XOR compare operation is effectively "OR"ed with the dynamic compare signal **1072** from the odd array using the complex OR-AND-Invert (OAI) gate **1065** which performs an OR-AND-INVERT function.

Insofar as additional tag bit column logic is implemented for the odd array (not shown) to generate the speculative odd miss signal **1072**, the TAG circuit includes selection circuitry for selecting between the speculative even and odd miss signals **1071**, **1072** based on the sum value (e.g., EA_{51}) that is computed for the least significant index bit position in Operands A and B (e.g., the sum of Operands A and B at bit position **51**). Though not shown, the sum value is computed by processing selected bits from Operands A and B (e.g., bits **52**-**63**) to generate a carry out value for the least significant

index bit position (e.g., by computing a carry out value for bit position **52**) which is then added to the least significant address bit (LSB bit **51**) from Operand A and the least significant address bit (LSB bit **51**) from Operand B to generate the sum value (EA_{51}) and optionally its complement.

The sum value (e.g., EA_{51}) is applied to inverter **1063** and OAI gate **1064**, which along with OAI gate **1065** and NAND gate **1066** effectively provide a multiplex or selection function to qualify the output from the speculative even compare **1071** and speculative odd compare **1072**, thereby choosing the miss/hit (match) signal **1067** from the odd or even address. If the sum value has a first value (e.g., logical "zero"), then the miss/hit signal **1071** from the even address array is chosen, and if the computed sum value has a second value (e.g., logical "one"), then the miss/hit signal **1072** from the odd address array is chosen. Thus, the sum value signal EA**51** which is the bit **51** of the output of the EA adder acts as a select signal for the multiplexer to choose between the speculative even or odd miss/hit signal to output the actual miss/hit signal.

To illustrate how the permission bitcells are arrayed and processed to identify valid speculative miss signals, reference is now made to FIG. **12** which depicts a circuit schematic diagram of the permission bit column logic **1200** in accordance with selected embodiments of the invention. The depicted permission bit column logic **1200** includes a plurality of tag bitcells (e.g., **1201**) arranged in a plurality of rows (e.g., **1210**, **1220**, **1230**) and columns (e.g., **1217**). Each bitcell (e.g., **1201**) may use substantially the same design and operation as shown in FIG. **8** so to generate a permission output signal **1206** in response to first and second control signals (CM, $\overline{CM}$) **1204**, **1205** and a compare wordline signal value (e.g., CWL**0**) for the permission bitcell.

The permission output signal **1206** is applied to the bitline driver output circuit block **1240** which is controlled by the precharge signal line **1241**. As shown, the permission output signal **1206** carry is coupled to a keeper pull-up circuit **1243**, **1244** for driving logical "one" values to a predetermined reference voltage (e.g., Vdd), where a first pull-up transistor **1243** is gated by an inverter **1244** which is coupled between the permission output signal line **1206** and the first pull-up transistor **1243**. In addition, the permission output signal line **1206** is coupled across a second pull-up transistor **1242** (e.g., PMOS transistor) to a predetermined reference voltage (e.g., Vdd) when the precharge signal line **1241** has a first logical state (e.g., "zero" volts). However, when the precharge signal line **1241** has a second logical state (e.g., "one" volt), the second pull-up transistor **1242** is not conductive, and the permission output signal **1206** is applied to an inverter circuit **1245** for purposes of generating an inverted permission output signal **1246**.

The inverted permission output signal **1246** is applied to the dynamic XOR logic circuit block **1250** which provides a dynamic comparator function. As depicted, the dynamic comparator block **1250** is controlled by the evaluate signal **1251** and mask signal **1252**, and controls the dynamic node **1274** (corresponding to the dynamic node **1055** depicted in FIG. **10**) to generate a speculative miss signal, even if all the tag bits on that row would otherwise generate a speculative hit signal. In operation, a logical "zero" stored in the permission bitcell **1201** will generate a logical "one" on the inverted permission output signal **1246**, thereby connecting the dynamic node **1257** to the reference supply voltage (e.g., ground) when the clock signal **1251** and the mask signal line **1252** are both set to logical "one" values. As described herein, the evaluate signal **1251** can be a clock signal or the delayed evaluate signal **1051** shown in FIG. **10**.

By now, it should be appreciated that there has been provided herein a method for generating a speculative hit/miss signal from a CAM array by placing a tag memory unit in parallel with an operand adder circuit to enable tag lookup and generation of speculative way hit/miss information directly from the operands without using the output sum of the operand adder. In selected embodiments, a method is provided in which base and offset operands are received, where each operand comprises a first plurality of address bits comprising tag bits and index bits. Each address bit from the base operand is paired with a corresponding address bit from the offset operand, thereby forming a plurality of index bit pairs and a plurality of tag bit pairs. The plurality of index and tag bit pairs are applied to an indexed content-addressable memory (CAM) array to generate two speculative miss signals from two speculatively indexed rows by using a two stage dynamic comparator to generate a speculative odd miss signal and a speculative even miss signal, respectively, in response to a delayed evaluate signal that is delayed with respect to a control word line signal applied to the two speculatively indexed rows. In selected embodiments, the bit pairs are applied by first generating PGZ access values for each of the index bit pairs. The generated PGZ access values include at least a P value (computed from a logical XOR operation performed on each of the paired bits), a G value (computed from a logical AND operation performed on each of the paired bits), and a Z value (computed from a logical AND operation performed on inverted versions of each of the paired bits). The PGZ access values are applied to a plurality of wordline generator circuits to speculatively index a possible odd memory array entry address from an odd CAM array and a possible even memory array entry address from an even CAM array. At the same time, PG comparison values are generated from the tag bit pairs, including a P value computed from a logical XOR operation on each tag bit pair, a complementary P value computed from a logical XNOR operation on each tag bit pair, and a G value computed from a logical AND operation on each tag bit pair. In parallel, a carry-out value is generated from at least the most significant bit of the base and offset operands. By applying the PG comparison values and the carry-out value to a content-addressable memory (CAM) array in parallel with the application of the PGZ access values, two speculative hit/miss signals are generated. In selected embodiments, this may be accomplished by applying the PG comparison values and carry-out value through the odd CAM array and even CAM array using a two stage dynamic comparator to generate a speculative odd miss signal and a speculative even miss signal, respectively, in response to a delayed evaluate signal. In a first comparison stage, PG comparison values are applied to each tag bitcell in any selected row of the even and odd CAM arrays, where each tag bitcell comprises embedded partial A+B=K comparison logic circuitry for generating required carry-in and produced carry-out values for each tag bitcell. This first stage comparison may implemented by applying a Generate (G) value, Propagate (P) value and a second data node in the CAM bitcell to a first logic circuit to generate a carry-out value (C_{out}), and applying a Propagate (P) value, complementary Propagate ($\overline{P}$) value, a first data node in the CAM bitcell, and a second data node in the CAM bitcell to a second logic circuit to generate a carry-in value (C_{in}). Subsequently, the required carry-in and produced carry-out values from each tag bitcell may be applied to an inverter circuit to generate, respectively, complementary carry-in value ($\overline{C_{in}}$) and a

complementary carry-out value ($\overline{C_{out}}$) for each tag bitcell. In a second comparison stage, the required carry-in and produced carry-out values (and their respective complements) for each tag bitcell are compared at a dynamic comparator to generate a speculative miss odd miss signal and speculative even miss signal by comparing the required carry-in value for each tag bitcell in a selected row with the produced carry-out value from the adjacent, less significant tag bitcell in response to a delayed evaluate signal. In other embodiments, the carry-out value and the PG comparison values may be applied by applying PG comparison values to tag bitcells in any selected row of the even and odd CAM arrays in a first comparison stage in response to a control wordline signal that is applied to select said row, where each tag bitcell comprises embedded partial A+B=K comparison logic circuitry for generating required carry-in and produced carry-out values for each tag bitcell. The control wordline signal is then applied to a delay circuit (e.g., a logical OR circuit with predetermined delay element that is connected to receive all control wordlines) to generate the delayed evaluate signal, and the required carry-in and produced carry-out values for each tag bitcell are compared at a dynamic comparator in a second comparison stage to generate a speculative miss odd miss signal and speculative even miss signal by comparing the required carry-in value for each tag bitcell in a selected row with the produced carry-out value from the adjacent, less significant tag bitcell in response to the delayed evaluate signal. Simultaneously or in parallel with the generation of the speculative odd miss signal and a speculative even miss signal, a sum value may be computed by adding at least the least significant index bit of the base and offset operands (plus any additional, less significant bits in the operands). The sum value is used to select one of the speculative hit/miss signals for output. The foregoing steps are performed concurrently while computing an effective address by adding the first and second operands in an operand adder circuit.

In another form, there is provided a data processing system and associated method of operation wherein one or more processors use selection circuitry to access a memory array having first and second speculatively indexed content addressable memory (CAM) arrays (e.g., an even CAM array and an odd CAM array) which each store a plurality of memory array entries. Each bitcell in the first and second speculatively indexed content CAM arrays includes embedded partial A+B=K compare logic to receive PGZ values generated from tag bits in the first plurality of base address bits and the first plurality of offset address bits, and to compute therefrom a carry out value and a carry in value for the bitcell in response to a control wordline signal. In selected embodiments, each bitcell includes a memory unit for storing a K value and a $\overline{K}$ value at first and second data nodes, and the embedded partial A+B=K compare logic includes a first comparison logic circuit for generating a carry out value for the bitcell by comparing the $\overline{K}$ value to a received G value and P value generated from tag bits in the first plurality of base address bits and the first plurality of offset address bits in response to a control wordline signal; and a second comparison logic circuit for generating a carry in value for the bitcell by comparing the K and $\overline{K}$ values to a received P value and $\overline{P}$ value generated from tag bits in the first plurality of base address bits and the first plurality of offset address bits in response to the control wordline signal. To generate complementary carry in and carry out values, there is provided a first inverter connected to receive the carry out value from each bitcell and to generate therefrom a complementary carry out value that is applied to the dynamic comparator circuit. In addition, there is provided a second inverter connected to receive the carry in value from each bitcell and to generate therefrom a complementary carry in value that is applied to the dynamic comparator circuit. The selection circuitry addresses a memory array entry from the memory array by applying to the memory array paired base and offset address bits that are not added together, using a dynamic comparator circuit to generate a speculative odd memory array entry address and a speculative even memory array entry address, respectively, in response to a delayed evaluate signal, and then selecting one of the speculative odd memory array entry address and the speculative even memory array entry address as the memory array entry for output. To this end, the selection circuitry includes a PGZ generator that generates PGZ values based upon the paired address bits wherein a P value results from a logical XOR operation on each address bit pair, a G value results from a logical AND operation on each address bit pair, and a Z value results from a logical AND operation performed on each inverted address bit pair. The selection circuitry also includes wordline generator circuits that evaluate PGZ values generated from index bits in the first plurality of base address bits and the first plurality of offset address bits to speculatively index a possible odd memory array entry address from the odd CAM array and a possible even memory array entry address from the even CAM array. In the selection circuitry, a first fast carry generation circuitry executes while the PGZ generator generates the PGZ values to generate a first carry value for the least significant tag bits in the first plurality of base address bits and the first plurality of offset address bits. In addition, a two-stage dynamic comparator circuit performs a speculative comparison on the possible odd memory array entry address and a possible even memory array entry address under control of a delayed evaluate signal. A signal generation circuit may be provided which receives a control wordline signal from each of the rows in the first and second speculatively indexed CAM arrays and generates therefrom the delayed evaluate signal. In selected embodiments, the dynamic comparator circuit performs the speculative comparison at each bitcell by comparing the carry in value from the bitcell with a carry out value from an adjacent less significant bitcell in response to the delayed evaluate signal to indicate a hit if the carry in value and carry out value match, and to indicate a miss if the carry in value and carry out value do not match. The selection circuit also includes an adder circuit for generating a sum value from at least the least significant index bits in the first plurality of base address bits and the first plurality of offset address. A match selector selects one of the possible memory array entry addresses based upon the sum value, and a latch latches the memory array entry corresponding to the selected memory array entry address. The data processing system also includes an operand adder circuit placed in parallel with the selection circuit for concurrently computing an effective address by adding the first and second operands.

In yet another form, there is provided a computer-implemented method for generating a miss signal from a CAM array. As the disclosed methodology, a base operand and offset operand are received, wherein each operand comprises a first plurality of tag and index address bits. Each tag and index address bits from the base operand is paired with a corresponding tag and index address bit from the offset operand, thereby forming a plurality of bit pairs. In addition, one or more PGZ values are generated for each bit pair by computing a P value from a logical XOR operation performed on each bit pair, a G value from a logical AND operation performed on each bit pair, a Z value from a

logical AND operation performed on inverted versions of each bit pair. PGZ values generated from paired index address bits are run through a plurality of wordline generators to speculatively index a possible odd memory array entry address from an odd CAM array and a possible even memory array entry address from an even CAM array. While the PGZ values are being generated, a fast carry generation is performed on a second plurality of bits which includes the index address bits from the base and offset operands to generate a first carry value for the least significant tag address bits in the base and offset operands. The first carry value is used to generate a speculative odd miss signal and a speculative even miss signal under control of a delayed evaluate signal by applying the first carry value, along with P values and G values generated from paired tag bits, to the speculatively indexed possible odd memory array entry address from an odd CAM array and the possible even memory array entry address from an even CAM array. In selected embodiments, the values are applied to the CAM array by comparing a K value stored in each bitcell of the speculatively indexed possible odd and even memory array entry addresses with a corresponding P value and G value generated from a corresponding tag address bit in the base and offset operands using partial A+B=K logic embedded in the bitcell to generate a carry out value and a carry in value for each bitcell under control of a control wordline signal, and then comparing the carry in value for each bitcell in a selected row with a carry out value from an adjacent, less significant bitcell in the selected row in response to a delayed evaluate signal. The carry in and carry out values from each bitcell may then be applied to inverters to generate, respectively, a complementary carry in value and complementary carry out value. In addition, the delayed evaluate signal may be generated from the control wordline signal. By computing a sum value for the least significant index bits in the base and offset operands while the PGZ values are being generated, one of the speculative odd miss signal and speculative even miss signal may be selected based upon the sum value.

The methods and systems for accessing a memory array circuit directly as shown and described herein may be implemented in hardware and/or software stored on a computer-readable medium and executed as a computer program on a general purpose or special purpose computer to perform certain tasks. In selected example implementations, the invention is implemented with hardware and software for accessing random access memory of the computer, including but not limited to hardware that is used to implement TAG circuit, fast carry generation and sum logic circuits, as well as software instructions or microcode used to implement the PGZ generation logic block. Until required by the computer, the set of instructions or microcode may be stored as instructions (program code) or other functional descriptive material in a code module in another computer memory, for example, in a hard disk drive, or in a removable memory such as an flash memory or optical disk (for eventual use in a CD ROM) or floppy disk (for eventual use in a floppy disk drive), or downloaded via the Internet or other computer network. In addition or in the alternative, a software implementation may be used, whereby some or all of the tag memory operations may be implemented with modules (e.g., procedures, functions, and so on) that perform the functions described herein. It will be appreciated that the separation of functionality into modules is for illustrative purposes, and alternative embodiments may merge the functionality of multiple software modules into a single module or may impose an alternate decomposition of functionality of modules. In any software implementation, the software code may be executed by a processor or controller, with the code and any underlying or processed data being stored in any machine-readable or computer-readable storage medium, such as an on-board or external memory unit. Thus, the present invention may be implemented as a computer program product for use in a computer in which the claimed functionality may be implemented in whole or in part as a state machine, firmware or software that is tangibly embodied in a computer readable, non-transitory storage medium.

Although the described exemplary embodiments disclosed herein are directed to various schemes for using a staged comparison with speculative indexed content addressable memory to generate speculative hit/miss signals, the present invention is not necessarily limited to the example embodiments which illustrate inventive aspects of the present invention that are applicable to a wide variety of latch devices, manufacturing methods and operational methodologies. Thus, the particular embodiments disclosed above are illustrative only and should not be taken as limitations upon the present invention, as the invention may be modified and practiced in different but equivalent manners apparent to those skilled in the art having the benefit of the teachings herein. Accordingly, the foregoing description is not intended to limit the invention to the particular form set forth, but on the contrary, is intended to cover such alternatives, modifications and equivalents as may be included within the spirit and scope of the invention as defined by the appended claims so that those skilled in the art should understand that they can make various changes, substitutions and alterations without departing from the spirit and scope of the invention in its broadest form.

Benefits, other advantages, and solutions to problems have been described above with regard to specific embodiments. However, the benefits, advantages, solutions to problems, and any element(s) that may cause any benefit, advantage, or solution to occur or become more pronounced are not to be construed as a critical, required, or essential feature or element of any or all the claims. The term "coupled," as used herein, is not intended to be limited to a direct coupling or a mechanical coupling. As used herein, the terms "comprises," "comprising," or any other variation thereof, are intended to cover a non-exclusive inclusion, such that a process, method, article, or apparatus that comprises a list of elements does not include only those elements but may include other elements not expressly listed or inherent to such process, method, article, or apparatus.

What is claimed is:

1. A method for generating a speculative miss signal from base and offset operands without requiring addition of the base and offset operands comprising:

receiving a base operand and offset operand, wherein each operand comprises a first plurality of address bits comprising tag bits and index bits;

pairing each address bit from the base operand with a corresponding address bit from the offset operand, thereby forming a plurality of index bit pairs and a plurality of tag bit pairs;

applying the plurality of index bit pairs and the plurality of tag bit pairs to an indexed content-addressable memory (CAM) array to generate two speculative miss signals from two speculatively indexed rows by using a two stage dynamic comparator to generate a speculative odd miss signal and a speculative even miss signal, respectively, in response to a delayed evaluate

signal that is delayed with respect to a control word line signal applied to the two speculatively indexed rows; and

selecting one of the speculative odd miss signal and speculative even miss signal for output based on a sum value computed by adding at least the least significant index bit of the base operand with the least significant index bit of the offset operand.

2. The method of claim **1**, where applying the plurality of index bit pairs and the plurality of tag bit pairs to the indexed CAM array comprises:

generating PGZ access values from the plurality of index bit pairs comprising a P value computed from a logical XOR operation on each index bit pair, a G value computed from a logical AND operation on each index bit pair, and a Z value computed from a logical AND operation performed on each inverted index bit pair;

applying the PGZ access values to a plurality of wordline generator circuits to speculatively index a possible odd memory array entry address from an odd CAM array and a possible even memory array entry address from an even CAM array;

generating PG comparison values from the plurality of tag bit pairs comprising a P value computed from a logical XOR operation on each tag bit pair, a complementary P value computed from a logical XNOR operation on each tag bit pair, and a G value computed from a logical AND operation on each tag bit pair;

generating a carry-out value from at least the most significant index bit of the base operand and the most significant index bit of the offset operand; and

applying the carry-out value and the PG comparison values through the odd CAM array and even CAM array using a two stage dynamic comparator to generate a speculative odd miss signal and a speculative even miss signal, respectively, in response to the delayed evaluate signal.

3. The method of claim **2**, where the carry-out value and the PG comparison values are applied in parallel with applying the PGZ access values.

4. The method of claim **2**, where generating PG comparison values occurs in parallel with generating the carry-out value.

5. The method of claim **1**, where applying the plurality of index bit pairs and the plurality of tag bit pairs to the indexed CAM array comprises:

generating PG comparison values from the plurality of tag bit pairs comprising a P value computed from a logical XOR operation on each tag bit pair, a complementary P value computed from a logical XNOR operation on each tag bit pair, and a G value computed from a logical AND operation on each tag bit pair;

applying PG comparison values to each tag bitcell in any selected row of the even and odd CAM arrays in a first comparison stage, where each tag bitcell comprises embedded partial A+B=K comparison logic circuitry for generating required carry-in and produced carry-out values for each tag bitcell;

applying the required carry-in and produced carry-out values generated from each tag bitcell to an inverter circuit to generate, respectively, complementary carry-in value ($\overline{C_{in}}$) and a complementary carry-out value ($\overline{C_{out}}$) for each tag bitcell; and

comparing the required carry-in value, produced carry-out value, complementary carry-in value ($\overline{C_{in}}$) and complementary carry-out value ($\overline{C_{out}}$) for each tag bitcell at a dynamic comparator in a second comparison stage to generate a speculative miss odd miss signal and speculative even miss signal by comparing the required carry-in value and complementary carry-in value ($\overline{C_{in}}$) for each tag bitcell in a selected row with the produced carry-out value and complementary carry-out value ($\overline{C_{out}}$) from the adjacent, less significant tag bitcell in response to the delayed evaluate signal.

6. The method of claim **5**, where applying PG comparison values to each tag bitcell comprises:

applying a Generate (G) value, Propagate (P) value and a second data node in the CAM bitcell to a first logic circuit to generate a carry-out value (C_{out}); and

applying a Propagate (P) value, complementary Propagate ($\overline{P}$) value, a first data node in the CAM bitcell, and a second data node in the CAM bitcell to a second logic circuit to generate a carry-in value (C_{in}).

7. The method of claim **1**, where applying the plurality of index bit pairs and the plurality of tag bit pairs to the indexed CAM array comprises:

generating a plurality of logical comparison values from the plurality of tag bit pairs;

applying the plurality of logical comparison values to tag bitcells in any selected row of the even and odd CAM arrays in a first comparison stage in response to a control wordline signal that is applied to select said row, where each tag bitcell comprises embedded partial A+B=K comparison logic circuitry for generating required carry-in and produced carry-out values for each tag bitcell;

applying the control wordline signal to a delay circuit to generate the delayed evaluate signal; and

comparing the required carry-in and produced carry-out values for each tag bitcell at a dynamic comparator in a second comparison stage to generate a speculative miss odd miss signal and speculative even miss signal by comparing the required carry-in value for each tag bitcell in a selected row with the produced carry-out value from the adjacent, less significant tag bitcell in response to the delayed evaluate signal.

8. The method of claim **7**, where applying the control wordline signal to the delay circuit comprises applying the control wordline signal to a logical OR circuit with predetermined delay element to generate the delayed evaluate signal.

9. The method of claim **1**, further comprising concurrently computing an effective address by adding the base and offset operands in an operand adder circuit.

10. A data processing system comprising:

one or more processors;

a memory array accessible by the one or more processors comprising first and second speculatively indexed content addressable memory (CAM) arrays comprising an even CAM array and an odd CAM array, each comprising a plurality of memory array entries; and

selection circuitry that addresses a memory array entry from the first and second speculatively indexed CAM arrays by applying paired base and offset address bits that are not added together to the memory array using a dynamic comparator circuit to generate a speculative odd memory array entry address and a speculative even memory array entry address, respectively, in response to a delayed evaluate signal, and then selecting one of the speculative odd memory array entry address and the speculative even memory array entry address as the memory array entry for output; and

an operand adder circuit placed in parallel with the selection circuit for concurrently computing an effective address by adding the base and offset address bits.

11. The data processing system of claim **10**, where the selection circuitry comprises:

a PGZ generator that generates PGZ values based upon paired based and offset address bits wherein a P value results from a logical XOR operation on each address bit pair, a G value results from a logical AND operation on each address bit pair, and a Z value results from a logical AND operation performed on each inverted address bit pair;

a plurality of wordline generator circuits that evaluate P values, G values, and Z values generated from index bits in the paired base and offset address bits to speculatively index a possible odd memory array entry address from the odd CAM array and a possible even memory array entry address from the even CAM array;

a first carry generation circuit that executes while the PGZ generator generates the PGZ values, the first carry generation circuit generating in a first carry value for the least significant tag bits in the paired base and offset address bits;

a two stage dynamic comparator circuit that performs a speculative comparison on the possible odd memory array entry address and a possible even memory array entry address under control of the delayed evaluate signal;

an adder circuit for generating a sum value from at least the least significant index bits in the paired base and offset address bits;

a match selector that selects one of the possible memory array entry addresses based upon the sum value; and

a latch that latches the memory array entry corresponding to the selected memory array entry address.

12. The data processing system of claim **10**, wherein the first and second speculatively indexed CAM arrays comprise one or more bitcells, each comprising:

a memory unit comprising two cross-coupled inverters coupled between first and second data nodes for storing a K value and a $\overline{K}$ value at the first and second data nodes;

a first comparison logic circuit for generating a carry out value for the bitcell by comparing the $\overline{K}$ value to a received G value and P value generated from tag bits in the first plurality of base address bits and the first plurality of offset address bits in response to a control wordline signal; and

a second comparison logic circuit for generating a carry in value for the bitcell by comparing the K and $\overline{K}$ values to a received P value and $\overline{P}$ value generated from tag bits in the first plurality of base address bits and the first plurality of offset address bits in response to the control wordline signal.

13. The data processing system of claim **12**, further comprising:

a first inverter connected to receive the carry out value from each bitcell and to generate therefrom a complementary carry out value that is applied to the dynamic comparator circuit; and

a second inverter connected to receive the carry in value from each bitcell and to generate therefrom a complementary carry in value that is applied to the dynamic comparator circuit.

14. The data processing system of claim **13**, where the dynamic comparator circuit compares the carry in value and complementary carry in value from each bitcell with a carry out value and complementary carry out value from an adjacent less significant bitcell in response to the delayed evaluate signal to indicate a hit if the carry in value and carry out value match, and to indicate a miss if the carry in value and carry out value do not match.

15. The data processing system of claim **10**, further comprising a signal generation circuit for receiving a plurality of control wordline signals from a corresponding plurality of rows in the first and second speculatively indexed CAM arrays and generating therefrom the delayed evaluate signal.

16. The data processing system of claim **10**, wherein the first and second speculatively indexed CAM arrays comprise one or more bitcells, each comprising:

a memory unit comprising two cross-coupled inverters coupled between first and second data nodes for storing a K value and a $\overline{K}$ value at the first and second data nodes; and

embedded partial A+B=K logic for comparing the K and $\overline{K}$ values to received P values and G values generated from tag bits in the first plurality of base address bits and the first plurality of offset address bits, and to compute therefrom a carry out value and a carry in value for the bitcell in response to a control wordline signal.

17. A method comprising:

receiving a base operand and offset operand, wherein each operand comprises a first plurality of tag and index address bits;

pairing each tag and index address bits from the base operand with a corresponding tag and index address bit from the offset operand, thereby forming a plurality of bit pairs;

generating one or more PGZ values for each bit pair wherein a P value results from a logical XOR operation performed on each bit pair, a G value results from a logical AND operation performed on each bit pair, a Z value results from a logical AND operation performed on inverted versions of each bit pair;

running P values, G values, and Z values generated from paired index address bits through a plurality of wordline generators to speculatively index a possible odd memory array entry address from an odd CAM array and a possible even memory array entry address from an even CAM array;

performing fast carry generation on a second plurality of bits comprising the index address bits from the base and offset operands to generate a first carry value for the least significant tag address bits in the base and offset operands;

generating a speculative odd miss signal and a speculative even miss signal under control of a delayed evaluate signal by applying the first carry value, along with P values and G values generated from paired tag bits, to the speculatively indexed possible odd memory array entry address from an odd CAM array and the possible even memory array entry address from an even CAM array;

computing a sum value for the least significant index bits in the base and offset operands while generating the speculative odd miss signal and the speculative even miss signal; and

selecting one of the speculative odd miss signal and speculative even miss signal based upon the sum value.

18. The method of claim **17**, where generating the speculative odd miss signal and the speculative even miss signal comprises:

comparing a K value stored in each bitcell of the speculatively indexed possible odd and even memory array entry addresses with a corresponding P value and G value generated from a corresponding tag address bit in the base and offset operands using partial A+B=K logic embedded in the bitcell to generate a carry out value and a carry in value for each bitcell under control of a control wordline signal; and

comparing the carry in value for each bitcell in a selected row with a carry out value from an adjacent, less significant bitcell in the selected row in response to a delayed evaluate signal.

19. The method of claim **18**, further comprising:

generating for each bitcell a complementary carry in value from the carry in value; and

generating for each bitcell a complementary carry out value from the carry out value.

20. The method of claim **18**, further comprising generating the generating the delayed evaluate signal from the control wordline signal.

* * * * *